VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2010 Jiří Dona

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Detekce a prevence anomálního chování v počítačových sítích

Detection and prevention of anomaly behaviour in computer networks

2009/2010 Jiří Dona

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Jiří Dona**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Detekce a prevence anomálního chování v počítačových sítích

Detection and prevention of anomaly behaviour in computer networks

Zásady pro vypracování:

Cílem diplomové práce je navrhnout a ověřit pomocí open source nástrojů detekci narušení síťového provozu.

1. Úvod do problematiky detekce a prevence narušení.
2. Návrh a praktická implementace vhodných nástrojů.
3. Ověření funkčnosti řešení pomocí různých penetračních testů.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

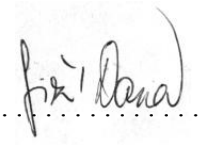
VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. 5. 2010 

podpis

Poděkování

Chtěl bych poděkovat panu Ing. Pavlovi Nevludovi za jeho vstřícný přístup a odbornou pomoc při psaní této diplomové práce.

Abstrakt

V této diplomové práci můžete najít základní rozdělení systémů pro detekci a prevenci narušení v síti. Dále pak postup vytváření pravidel pro program SNORT v režimu síťového detektoru narušení (IDS). Tato pravidla jsou názorně předvedeny a rozebrány. Součástí práce je také popis použití nástrojů pro otestování vytvořených pravidel. Jedná se o generátory provozu hping, nemesis a packETH.

Klíčová slova

SNORT, pravidla, IDS, IPS, penetrační nástroj, generátor provozu, hping, nemesis, packeth, zabezpečení sítě

Abstract

In this graduation thesis, you can find basic divisions of intrusion detection and prevention systems in computer networks. Next you can learn about creating rules for programme SNORT in intrusion detection mode (IDS). These rules are clearly shown and described. In this thesis are included descriptions of programmes for testing rules in IDS. That are network generators hping, nemesis and packETH.

Key words

SNORT, rules, IDS, IPS, penetration tool, network generator, hping, nemesis, packeth, network security

Seznam použitých symbolů a zkratek

ARP	protokol pro zjištění MAC adresy podle IP adresy (address resolution protocol)
BSD	Unix vytvořený Kalifornskou univerzitou v Berkeley (Barkley Software Distribution)
CIDR	beztřídní přidělování neveřejných IP adres (Classless Inter-Domain Routing)
CUI	příkazový řádek (Command Line Interface)
FTP	protokol pro přenos souborů (File Transfer Protocol)
GRE	tunelovací protokol (Generic Rating Encapsulation)
GUI	grafické rozhraní (Graphical User Interface)
IANA	organizace pro přidělování IP adres na Internetu (Internet Assigned Numbers Authority)
ICMP	protokol kontrolních zpráv (Internet Control Message Protocol)
IDS	systém detekce narušení (intrusion detection system)
IPS	systém prevence narušení (intrusion prevention system)
IPv4	internetový protokol verze 4
IPv6	internetový protokol verze 6
IPX	síťový protokol používaný v operačních systémech Novell NetWare (Internetwork Packet Exchange)
OS	operační systém
OSPF	routovací protokol (Open Shortest Path First)
PC	uživatelský počítač (personal computer)

POP3	protokol pro stahování e-mailů (Post Office Protocol version 3)
RFC	označení řady standardů popisujících internetové protokoly (request for comments)
RIP	routovací protokol (Routing Information Protocol)
RSPAN	metoda zrcadlení portů z více cisco přepínačů (Remote Switched Port Analyzer)
SMTP	protokol pro odesílání e-mailů (Simple Mail Transfer Protocol)
SPAN	metoda zrcadlení portů z jednoho cisco přepínače (Switched Port Analyzer)
TCP	síťový protokol transportní vrstvy (Transmission Control Protocol)
UDP	síťový protokol transportní vrstvy (User Datagram Protocol)

Obsah

1. Úvod

V úvodu této práce bych rozebral samotné zadání. Nejprve jsou to detekce a korekce, tyto dvě slova je určitě dobré umět rozlišit. Jestliže něco detekuji, jedná se o pasivní úkon, tedy o jakési oznámení možného narušení sítě. Zatímco korekce je aktivní proces, nebude pouze informovat, ale zasáhne konkrétním způsobem, například zablokuje datový tok z podezřelé IP adresy. Dále se v zadání objevuje anomální chování v počítačových sítích. Pod tímto pojmem bychom si měli představit jakékoliv narušení, které může ohrozit bezpečnost sítě a umožnit tak ztrátu či odcizení citlivých dat.

Síťová bezpečnost je v současné době velmi diskutovaným tématem. S rostoucím počtem uživatelů a jejich technickou zdatností roste také riziko narušení sítí. Proto se v této práci zmiňuji o komplexní ochraně sítí, nejvíce se pak zaměřuji na systémy IDS.

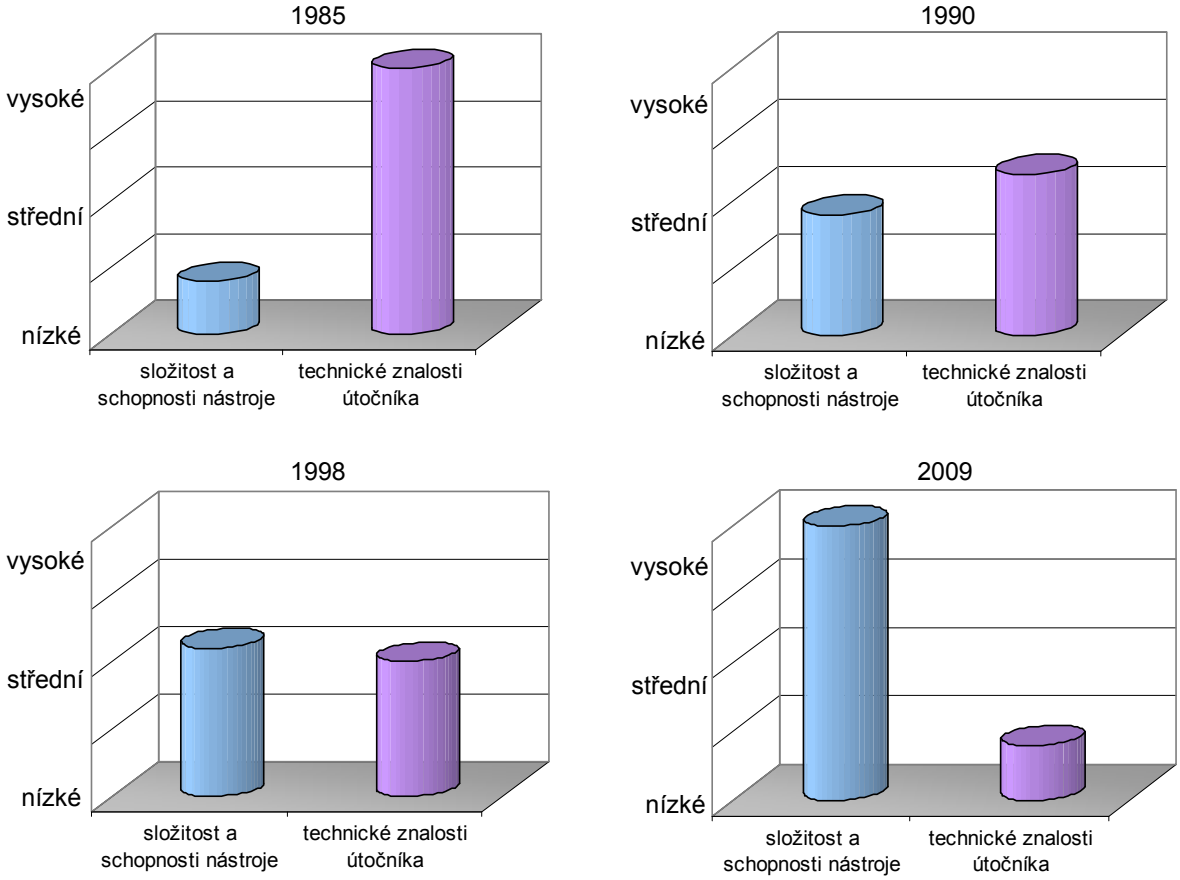
Jelikož je mým úkolem využít open source nástroj pro detekci narušení síťového provozu, volil jsem software SNORT od společnosti Sourcefire®. Ten jsem instaloval na operační systém Linux Ubuntu 9.10. Program SNORT umí pracovat ve třech základních režimech. Je to režim sledování provozu, režim ukládání systémových zpráv a síťový detektor narušení (IDS). Od verze 2.3.0 navíc podporuje režim inline, neboli IPS. V režimech IDS a IPS mám možnost vytvářet pravidla, podle kterých se SNORT rozhodne, zda jde o případnou hrozbu či nikoliv.

Ve chvíli, kdy takový detektor narušení implementuji do své sítě, je důležité systém otestovat. Tímto jsem se zabýval v poslední kapitole této diplomové práce. Penetrační nástroje pak umožňují generovat pakety s přesně definovanými parametry, které potřebuji ke zjištění účinnosti IDS systému. Kromě testování pomocí paketových generátorů jsem IDS systém vyzkoušel také v reálném provozu. K tomu jsem využil přenos souborů z FTP serveru a rozesílání emailů s různým obsahem pomocí poštovního SMTP a POP3 serveru.

2. Zabezpečení počítačových sítí

2.1. Historie

V dnešní době je zabezpečení sítě často diskutovanou otázkou. S rostoucím množstvím prostředků k prolomení zabezpečení a navyšujícím se počtem lidí znalých práce s počítačem na vyšší úrovni je stále těžší naší síť uchránit. Nástrojů, kterými je možno síť zabezpečit dnes existuje celá řada. V každé síti by pak určitě neměl chybět firewall, antivir a IDS či IPS systém. Všechny tyto prvky tvoří jakousi komplexní ochranu zabezpečené sítě. Při implementaci těchto prvků je důležité neusnout na vavřínech a sledovat dál vývoj nových aktualizací, hrozeb a možností útočníků. Jelikož jak se vyvíjí možnosti ulehčující práci uživatelům, tak stejně rychle ne-li rychleji se vyvíjí nové možnosti pro útočníky, kteří už dnes nemusí mít znalosti v oblasti počítačových sítí na takové úrovni, jak tomu bylo jen pár let dříve. Tato skutečnost je názorně předvedena na následujících grafech (Graf 2.1.1).

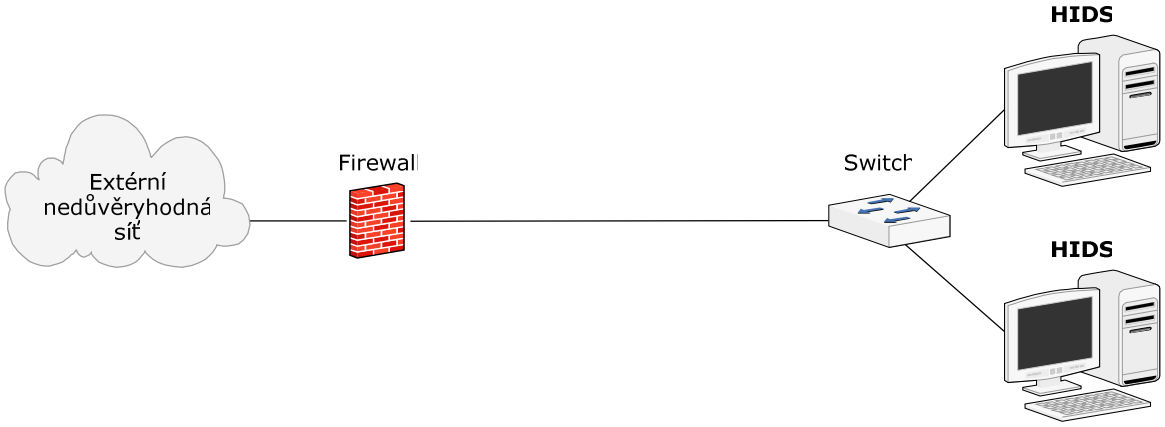


Graf 2.1.1 Vývoj míry potřebných znalostí k útoku na bezpečnost sítě

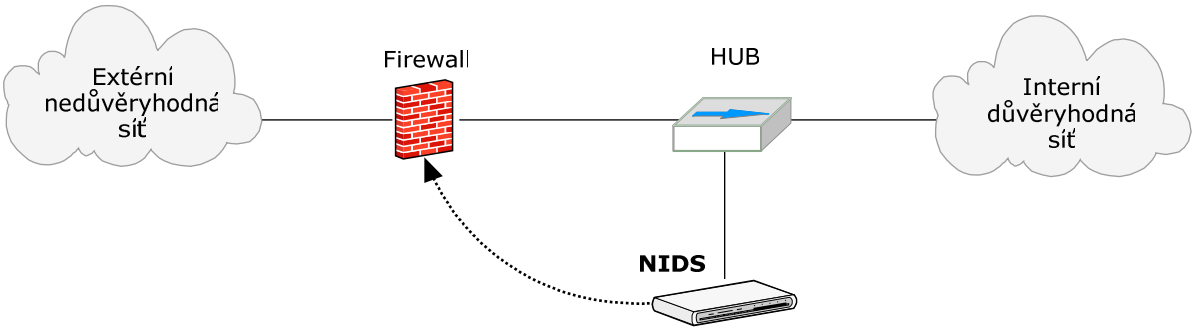
Z těchto grafů (Graf 2.1.1) vyplývá, že kolem roku 1985, kdy se počítačové sítě teprve rozvíjely, bylo zapotřebí hodně znalostí a sofistikovanost softwarových prostředků byla minimální, zatímco již o pět let později se tyto hodnoty vyrovnaly. V dnešní době třeba opravdu minimum znalostí v oblasti počítačových sítí, jelikož propracovanost softwaru použitelného pro vniknutí do cizí sítě, případně softwaru pro odcizení osobních dat, je na vysoké úrovni. Příkladem může být aplikace pro záznam zadaných kláves na počítači. Po instalaci takového programu může útočník získat přístupové hesla k bankovním účtům, firemním databázím atd.

2.2. Detekce narušení (IDS)

Nově se rozvíjejícím prvkem na poli bezpečnosti sítí jsou systémy pro detekci narušení sítě (Network Intrusion Detection Systems) označované jako NIDS nebo obecněji IDS. Obecněji proto, že se tyto IDS systémy dělí na NIDS a HIDS (Hostbased Intrusion Detection Systems). HIDS jsou pak nainstalovány přímo u koncového uživatele, zatímco NIDS se zpravidla umisťují za firewall směrem do vnitřní sítě. V této práci se zabývám zejména NIDS systémy. Hned v úvodu bych měl určitě zmínit, že IDS systém je pasivním prvkem v síti. Jeho úkolem je pouze zaznamenat útok na interní síť a oznámit tuto skutečnost administrátorovi. IDS systémy jsou i přesto schopny jistého aktivního zákroku. Jedná se především o předání informací, ohledně zjištěných útoků, firewallu a ten následně může dynamicky změnit části své politiky tak, aby zamezil přenosu vyhodnoceného jako útok.

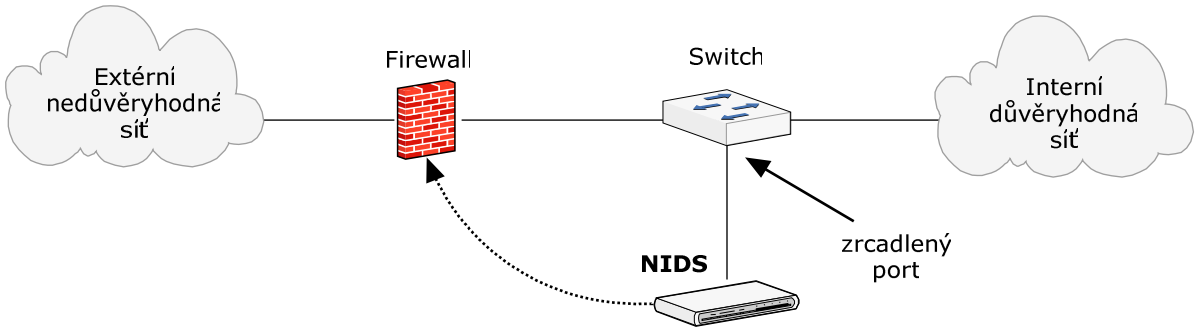


Obr. 2.2.1 Schéma Hostbased IDS



Obr. 2.2.2 Schéma Network IDS v zapojení s rozbočovačem

Ukázka HIDS systému je na obrázku 2.2.1 (Obr. 2.2.1), na uživatelském počítači ve vnitřní síti je nainstalován software pro detekci narušení a hlídá možné hrozby z externí sítě. Nevýhodou je, že chráněna bude pouze jedna koncová stanice, zatímco v případě NIDS (Obr. 2.2.2) sledujeme veškerý provoz přicházející do naší interní sítě případně z interní sítě ven. Způsobů jak umístit síťový detektor narušení do vlastní sítě je více. Konkrétně jsou to tyto tři, pomocí rozbočovače neboli hubu, jak tomu je na obrázku výše (Obr. 2.2.2). Druhou možností jak NIDS zapojit je pomocí přepínače neboli switche, na něm je pak zapotřebí nastavit zrcadlení portů (často se v literatuře vyskytuje pod anglickým názvem *port mirroring*), tak aby se k IDS dostal veškerý provoz procházející do, případně z, interní zabezpečené sítě. Ukázka zapojení je na obrázku níže (Obr. 2.2.3).



Obr. 2.2.3 Schéma síťového IDS v zapojení s přepínačem

Příklad nastavení zrcadlení portů uvádím pro cisco zařízení, u ostatních výrobců bude princip velmi podobný. Metoda zrcadlení portů (port mirroring) je u cisca pojmenována jako Switched Port Analyzer (SPAN) nebo Remote Switched Port Analyzer (RSPAN). SPAN pracuje v rámci jednoho switche, kde se zdrojový (monitorovaný) port, výběr více portů nebo virtuální LAN (VLAN), zrcadlí na cílový (monitorující) port. Platí, že pokud nastavím port jako monitorující, tak zablokuji připojenému zařízení (v našem případě IDS) zpětnou komunikaci přes toto rozhraní, pouze se zde bude kopírovat zvolený provoz. Při konfiguraci SPAN je zapotřebí nejdříve nadefinovat monitorovací relace (neboli session). Relace je označena číslem od 1 do 66 a na každý přepínač lze nadefinovat maximálně dvě takovéto relace. Jako zdroj (source) mohu zadat rozhraní (interface) nebo vlan, případně několik rozhraní či vlanů. Není ovšem možno kombinovat rozhraní a vlany v rámci jedné relace. Jako cíl (destination) zadám jedno, případně více rozhraní

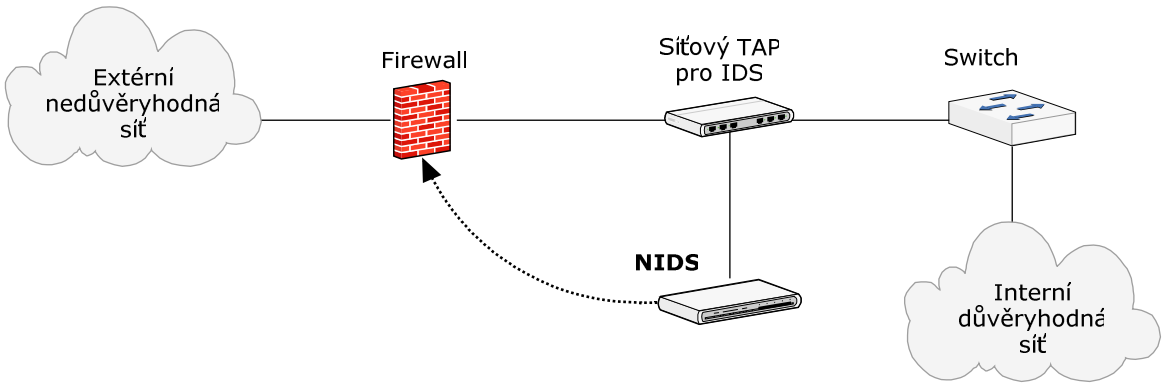
(portů). Při zadávání zdrojových portů mohu navíc určit, jestli se bude zrcadlit příchozí provoz (rx), odchozí provoz (tx) nebo můžeme využít defaultního nastavení, což je zachytávaní obou směrů. Na výpise (Výpis 2.2.4) je uveden příklad konfigurace.

```
Switch(config)# monitor session 1 source interface fa 0/15 rx
Switch(config)# monitor session 1 destination interface fa 0/24
```

Výpis 2.2.4 Konfigurace zrcadlení portů v cisco IOS

RSPAN se od SPAN liší v tom, že může zrcadlit porty z více switchů. Konfigurace RSPAN je složitější než v případě SPAN. Více informací naleznete v literatuře [16].

Posledním třetím způsobem je zapojení s využitím síťového TAPu. Jedná se o virtuální ethernet spojení. Zařízení, které vzájemně komunikují přes TAP rozhraní, se chovají, jako by byly ve společné lokální síti. Schéma zapojení je uvedeno na následujícím obrázku (Obr. 2.2.5). Výhoda TAPu tkví v tom, že není viditelné v síti, jako třeba přepínač, proto se jedná o bezpečný prvek z hlediska napadnutelnosti. Další výhodou je, že dokáže propustit provoz i v případě výpadku napájení, neztratíme tedy konektivitu. Možnou nevýhodou může být jen cena, která bývá v případě TAPu, od společnosti comcraft, vyšší než u přepínače se stejným počtem portů.



Obr. 2.2.5 Schéma síťového IDS v zapojení s TAP rozhraním

IDS systém sleduje provoz procházející mezi externí a interní sítí, v případě podezřelého provozu administrátora upozorní varovnou zprávou. Detekční systémy mohu dále rozdělit podle toho, jakým způsobem se rozhodují, zda se jedná o podezřelý provoz či nikoliv. První způsob je založen na sledování konkrétních parametrů paketů, může to být typ protokolu (TCP, ICMP…),

obsah hlaviček paketů, čísla portů, IP adresy atp. O IDS pracujících na tomto principu se hovoří jako o systémech založených na signaturách. Druhý způsob IDS je postaven na myšlence sledovat statistiku toku dat, vytvořit si model toho, co je v síti běžný provoz a v případě, že se bude ze statistického hlediska vymykat, ohlásí potencionální narušení sítě. Takový princip se nazývá detektor anomálního chování. Existuje ještě třetí způsob rozhodování a ten je založen na detekci odchylek od protokolových norem. Vychází z definic jednotlivých protokolů ve standardech RFC. Veškeré odchylky od těchto norem jsou pak dále analyzovány a vyhodnocovány. Z pohledu přesnosti objevení útoku je nejpřesnější systém založený na signaturách, tedy na prohledávání shody parametrů paketů. Nevýhodou ovšem je, že se dá použít pouze na odhalení již známých signatur útoku. Naopak druhý uváděný způsob, detektor anomálního chování, dokáže odhalit i doposud nepopsaný způsob útoku. Jeho nevýhodou je větší množství falešně pozitivních zpráv o útoku.

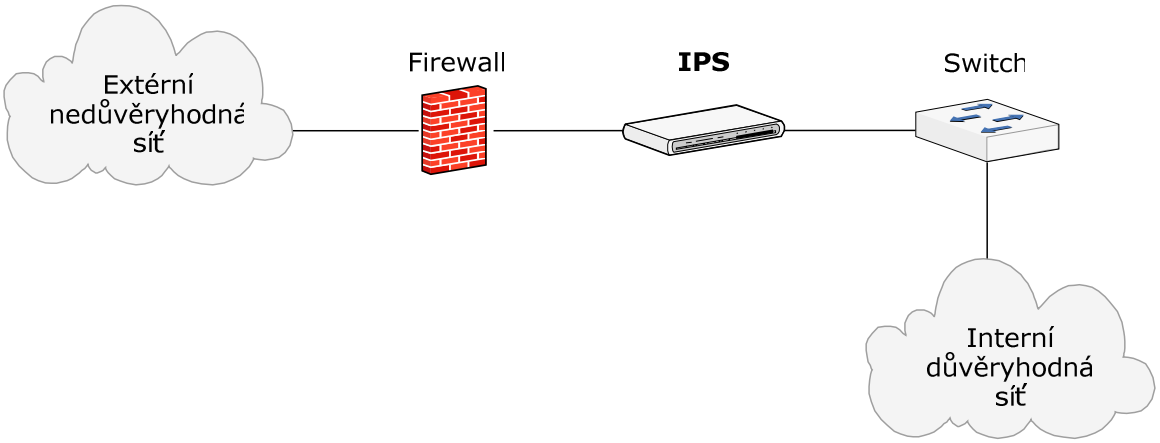
Zde by bylo vhodné uvést, jaké jsou možnosti ohlášení útoků u IDS a IPS systémů. Jde o to, že ne vždy ohlášené napadení naší sítě je oprávněné a ne na všechny útoky nám detekční nebo prevenční systém zareaguje. Jaké možnosti tedy existují? Jsou to tyto čtyři.

- **True positive (opravdu pozitivní)** – oprávněný útok, který byl zachycen IDS systémem a vyvolá poplach
- **False positive (falešně pozitivní)** – událost, jež spustí poplach u IDS systému, i když k útoku nedošlo
- **False negative (falešně negativní)** – selhání IDS systému, který neohlásí uskutečněný útok
- **True negative (opravdu negativní)** – nedochází k žádnému poplachu ve chvíli, kdy nedochází k útoku

Na základě těchto možností se následně provádí „alarm filtering“ neboli filtrace nahlášených, opravdu pozitivních a falešně pozitivních poplachů, aby se potvrdily, případně vyvrátily.

2.3. Prevence narušení (IPS)

V předchozí podkapitole jsem psal o nově se rozvíjejícím prvku v síťové bezpečnosti v podobě IDS. IPS systém opět sleduje provoz v síti a detekuje útok s tím rozdílem, že IPS dokáže podniknout konkrétní kroky k zamezení útoku nebo k jeho přerušení samostatně. Může například zablokovat provoz z podezřelé IP adresy či portu. Jedná se tedy o aktivní prvek v síti. Schéma zapojení je naznačeno na obrázku 2.3.1 (Obr. 2.3.1).

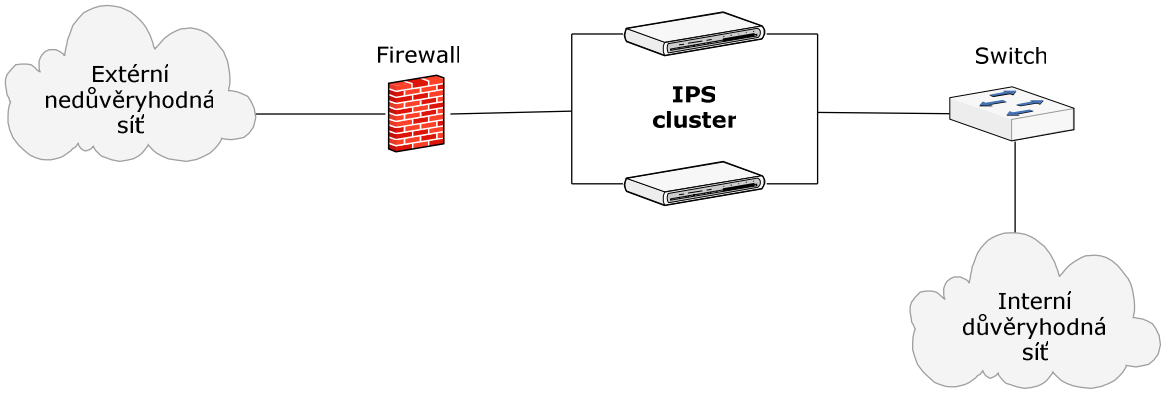


Obr. 2.3.1 Schéma zapojení IPS

Jednoznačnou výhodou takového IPS systému je jeho rychlost, která je ve srovnání s rychlostí reakce administrátora (u IDS systému) téměř okamžitá. Na druhou stranu, jestliže IDS systém vyhodnotí v komunikaci opravdu pozitivní útok a vyvolá poplach, který administrátor vyhodnotí jako falešně pozitivní, uživatelé v interní síti nic nepoznají. IPS systém by ve stejné situaci například zablokoval provoz určité skupině uživatelů nebo by znepřístupnil služby vázané k externí IP adrese nebo portu. Mohl by tak způsobit ztrátu konektivity k internetu či nefunkční tisk na síťových tiskárnách. To už by ovšem uživatelé interní sítě pocítili. Proto může tento automatický zásah způsobit nemalé potíže. Z tohoto důvodu je také kladen daleko větší důraz na nastavení IPS systémů tak, aby produkovaly co nejmenší počet falešně pozitivních poplachů, než u systémů IDS.

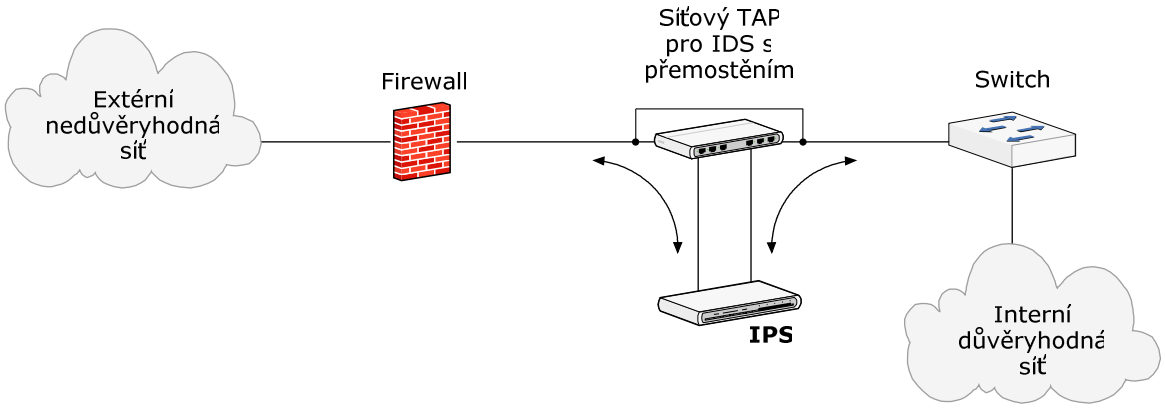
Jelikož se IPS systémy zapojují do sítě sériově, může se stát, že dojde ke ztrátě konektivity s externí sítí vlivem poruchy IPS zařízení, jeho zahlcením, nebo ztrátou napájení. Toto riziko se dá samozřejmě snížit pravidelnou údržbou, kontrolou funkčnosti a záložním zdrojem elektrické energie, ale pořád je zde vysoké riziko ztráty konektivity. Zvýšení spolehlivosti se proto řeší

řazením IPS systémů do takzvaných clusterů neboli paralelním seskupením více IPS (standardně se zapojují dva). Příklad zapojení je pak uveden na následujícím obrázku (Obr. 2.3.2).



Obr. 2.3.2 Schéma zapojení IPS s využitím clusteru

Druhou možností je využití síťového TAPu podobně, jako u IDS systému. Ale tentokrát musíme zajistit, aby provoz neprocházel přímo z externí do interní sítě a naopak, ale aby se provoz převáděl přes IPS systém. Síťový TAP navíc vytváří bypass neboli přemostění, které se stává aktivním právě v případě nefunkčnosti IPS zařízení. Schéma zapojení IPS se síťovým TAPem je na obrázku níže (Obr. 2.3.3).



Obr. 2.3.3 Schéma zapojení IPS s využitím síťového TAPu

V dnešní době se naskýtá ještě poslední možnost a to je využití IPS zařízení, která mají tento bypass integrovaný v sobě a při výpadku napájení nebo nějaké interní poruše zajistí propustnost dat.

V úvodu této práce jsem zmiňoval komplexní ochranu sítě. Kombinací detekčního nebo prevenčního systému a firewallu jsem schopen zajistit vysoký stupeň zabezpečení před neoprávněným vniknutím narušitele do interní sítě. Samozřejmostí je provádět pravidelné aktualizace vědomostní databáze detekčního nebo prevenčního systému tak, abych byl schopen odolávat i těm nejaktuálnějším hrozbám ze strany útočníků.

3. Open source aplikace SNORT

Při výběru IDS systémů máme k dispozici celou řadu možností, některé jsou placené, některé jsou volně ke stažení a jiné jsou na bázi otevřeného zdrojového kódu. Právě poslední možností je aplikace SNORT od autora jménem Marty Roesch a společnosti Sourcefire®. Možná právě díky tomu, že se jedná o zdrojově otevřený systém, který si jeho uživatelé postupem času rozšířili a vyladili téměř k dokonalosti, se stal jedním z nejrozšířenějších řešení pro IDS systémy. Více informací o programu SNORT naleznete na jeho internetových stránkách http://www.snort.org a v literatuře [9]. Výhoda SNORTu je určitě v tom, že je k dispozici pro různé operační systémy jako jsou Linux, AIX, BSD, HP/UX, IRIX, Mac OS X, Solaris, Tru64 a také Microsoft Windows. Další výhodou je schopnost vykonávat analýzu provozu v reálném čase. SNORT je používán spíše jako NIDS (detektor síťového narušení), ale pokud je dobře nakonfigurován, dá se použít i jako HIDS (detektor narušení hostitelského počítače). SNORT při odchytávání paketů využívá knihovnu libpcap a pomocí promiskuitního módu může zachytávat veškeré pakety, jež se v síti vyskytují. Zachycené pakety následně dekóduje a s využitím definovaných pravidel je testuje. V případě prokázání podezřelého provozu pak administrátora informuje varovnou zprávou.

SNORT umí pracovat ve třech základních režimech. Je to režim sledování provozu (sniffer), režim ukládání systémových zpráv (logger) a režim síťového detektoru narušení (IDS). Čtvrtým rozšiřujícím režimem je inline mód (IPS). Nyní už vám tedy přiblížím, jak se SNORT ve výše uvedených režimech chová a nastavuje.

3.1. Režim sledování provozu

V režimu sledování provozu (snifer) SNORT pouze odposlouchává provoz v síti, sleduje data obsažená v hlavičce a těle všech procházejících paketů, který následně vypisuje na obrazovku. Při spouštění SNORTu v režimu sledování provozu máme tři možnosti. První možnost je zadáním příkazu `Snort -dv`. Na obrazovce se vypíší informace o třetí a čtvrté vrstvě OSI modelu. Výstup pak může být následující (Výpis 3.1.1).

```
root@student-desktop:~# snort -dv
04/17-23:28:38.961141 10.0.2.15:555 -> 192.168.1.20:666
①TCP TTL:255 TOS:0x0 ID:4660 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x15  Ack: 0x14  Win: 0xFA0  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
04/17-23:28:40.961257 10.0.2.15 -> 192.168.1.20
②ICMP TTL:255 TOS:0x0 ID:4660 IpLen:20 DgmLen:28 DF
Type:8  Code:0  ID:1   Seq:1  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Výpis 3.1.1 výstup Snortu po zadání příkazu snort -dv

V první části výstupu ① si můžete všimnout zachycení TCP paketu a v druhé části ② je zaznamenán ICMP paket. U obou částí výpisu pak naleznete položky třetí vrstvy OSI modelu, například jsou to zdrojové a cílové IP adresy, protokoly, TTL atd. I prvky čtvrté vrstvy jako zdrojový a cílový port, sekvenční číslo, číslo potvrzení nebo typ a kód ICMP zprávy. Jestliže bych chtěl vypsat informace o druhé vrstvě OSI modelu (MAC adresy), zadal bych do příkazu `snort -ev` (Výpis 3.1.2) nebo kombinací všech tří tedy `snort -dve`. Zápis je možný i odděleně `snort -d -v -e` výstup bude stejný.

```
root@student-desktop:~# snort -ev
04/17-23:30:54.158367 8:0:27:6A:CA:FB -> 0:40:F4:92:E9:91 type:0x800 len:0x36
10.0.2.15:555 -> 192.168.1.20:666 TCP TTL:255 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
DF
***A**** Seq: 0x15  Ack: 0x14  Win: 0xFA0  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Výpis 3.1.2 Výpis Snortu po zadání přkazu snort –ev

3.2. Režim ukládání systémových zpráv

Funkce režimu ukládání systémových zpráv (logger) spočívá v tom, že se data zachycená programem SNORT nevypisují na obrazovku, jak tomu bylo v případě režimu sledování provozu, ale ukládají se do souboru na pevném disku. Nyní už přejdu ke konkrétní ukázce jak zapnout režim ukládání systémových zpráv. Provedu to zadáním následujícího příkazu (Výpis 3.2.1)

`snort -l /etc/snort/logovana`

Výpis 3.2.1 Příkaz pro uložení do složky na pevném disku

SNORTu tím sdělím, aby zachytil druhou třetí i čtvrtou vrstvu OSI modelu a za `-l` uvádím cestu složky do které se bude záznam ukládat, tedy do složky „logovana“. V případě že mu ovšem neuvedu žádný konkrétní rozsah adres, bude zachytávat pouze pakety z naší lokální sítě. Jednoduchou úpravou příkazu uvedeného výše (Výpis 3.2.1) docílím záznamu paketů z konkrétního rozsahu adres, navíc přidám argumenty `-dev`, aby se mi zachycené pakety rovněž vypsaly na obrazovku. Upravený příkaz by tedy vypadal následovně (Výpis 3.2.2).

`snort -dev -l /etc/snort/logovana -h 10.0.2.0/24`

Výpis 3.2.2 Příkaz pro uložení záznamu o provozu z konkrétního rozsahu adres

Zde jsem přidal argument `-h` a za ním následuje rozsah adres, v tomto případě 10.0.2.0/24 tedy adresy od 10.0.2.0 do 10.0.2.255. Ve složce etc/snort/logovana se pak vytvoří samostatný soubor pro každou adresu. V následujícím textu je zachycený celý výpis SNORTu (Výpis 3.2.3) od jeho spuštění až po jeho ukončení, takže si můžete udělat představu o tom, jaké informace nám SNORT poskytne.

```
root@student-desktop:~# snort -dev -l /etc/snort/logovana -h 10.0.2.0/24
① Running in packet logging mode
Log directory = /etc/snort/logovana

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.8.4.1 (Build 38)
   ''''    By Martin Roesch & The Snort Team: http://www.Snort.org/team.html
           Copyright (C) 1998-2009 Sourcefire, Inc., et al.
           Using PCRE version: 7.8 2008-09-05

Not Using PCAP_FRAMES
04/17-23:22:40.992588 8:0:27:6A:CA:FB -> 0:40:F4:92:E9:91 type:0x800 len:0x36
② 10.0.2.15:555 -> 192.168.1.20:666 TCP TTL:255 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
DF
***A**** Seq: 0x15  Ack: 0x14  Win: 0xFA0  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
  04/17-23:22:42.992696 8:0:27:6A:CA:FB -> 0:40:F4:92:E9:91 type:0x800 len:0x2A
③ 10.0.2.15 -> 192.168.1.20 ICMP TTL:255 TOS:0x0 ID:4660 IpLen:20 DgmLen:28 DF
  Type:8  Code:0  ID:1   Seq:1  ECHO

  =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

  ^C*** Caught Int-Signal
  Run time prior to being shutdown was 13.534247 seconds
  ===============================================================================
  Packet Wire Totals:
     Received:           2
     Analyzed:           2 (100.000%)
④     Dropped:           0 (0.000%)
  Outstanding:           0 (0.000%)
  ===============================================================================
  Breakdown by protocol (includes rebuilt packets):
        ETH: 2          (100.000%)
        IP4: 2          (100.000%)
⑤       TCP: 1          (50.000%)
        UDP: 0          (0.000%)
       ICMP: 1          (50.000%)
      Total: 2
  ===============================================================================
  Action Stats:
  ALERTS: 0
⑥ LOGGED: 2
  PASSED: 0
  ===============================================================================
  Snort exiting
```

Výpis 3.2.3 Výpis SNORTu v režimu ukládání systémových zpráv

Abych se ve výpisu (Výpis 3.2.3) vyznal, popsal jsem jeho zajímavé části číslem. Pod číslem jedna ① je zadán příkaz shodný s příkazem ve výpisu 3.2.2 (Výpis 3.2.2). Program SNORT vypíše režim, ve kterém jsem jej spustil a cestu ke složce, do které bude ukládat informace o zachyceném provozu. U čísla dvě ② už program zachytil TCP paket a u čísla tři ③ se jedná o ICMP paket. Poté jsem program ukončil a SNORT vypsal statistiku o zaznamenaném provozu. ④▪označuje statistiku počtu příchozích (received) paketů a počet analyzovaných (analyzed) paketů, v obou případech jsou to pakety dva. Pod pětkou ⑤ je statistika podle protokolu, jeden paket byl

TCP, druhý ICMP a oba by spadaly do IPv4 a ethernetového rámce. U posledního čísla tedy šest ⑥ je pak statistika akcí, v tomto případě byly dva pakety uloženy jako log.

Jak jsem již říkal, tyto výpisy jsou uloženy do námi definované složky, a pro každou IP adresu zvlášť. V případě, že je budeme chtít zpětně zkontrolovat, či nějak dále analyzovat, můžeme si je zpětně vypsat na obrazovku. Já při tomto úkonu vždy postupoval tak, že jsem si nejprve vypsal soubory uložené v mé logovací složce pomocí příkazu `dir /etc/snort/logovana` (Výpis 3.2.4).

```
root@student-desktop:~# dir /etc/snort/logovana
Snort.log.1271522712  Snort.log.1271529104  Snort.log.1271531643
Snort.log.1271522775  Snort.log.1271529228  Snort.log.1271538097
Snort.log.1271522821  Snort.log.1271531619  Snort.log.1271539355
```

Výpis 3.2.4 Příkaz pro vypsání obsahu složky

K otevření konkrétního souboru (například Snort.log.1271522712) jsem použil argument `-r` v praxi je pak použit ve výpisu 3.2.5 (Výpis 3.2.5).

```
snort -dve -r /etc/snort/logovana/Snort.log.1271522712
```

Výpis 3.2.5 Příkaz pro vypsání zaznamenaného provozu

Mohu také použít filtr a nechat si tak vypsat kupříkladu pouze TCP pakety, ukázka takového příkazu je ve výpisu 3.2.6 (Výpis. 3.2.6).

```
snort -dve -r /etc/snort/logovana/Snort.log.1271522712 tcp
```

Výpis 3.2.5 Příkaz pro vypsání zaznamenaného provozu s filtrem

3.3. Režim inline (IPS)

Tento režim SNORT podporuje od verze 2.3.0 RC1 a jedná se tedy o prevenční systém. Jelikož je tato práce zaměřena na praktickou ukázku detekčních systémů, uvádím tuto kapitolu spíše jako doplnění režimů programu SNORT.

V režimu IPS SNORT získává Pakety z iptables místo z libpcap a poté využívá nové typy pravidel pro propuštění nebo zahazování paketů. V případě že bych chtěl SNORT využívat v inline režimu, musel bych nejdříve stáhnout a zkompilovat iptables kód, který by současně zavedl libipq

knihovnu, jež umožní SNORTu v inline režimu styk s iptables. Nakonec bych musel nainstalovat LibNet. Informace o tom, kde tyto knihovny stáhnout naleznete v literatuře [9]. Nyní bych uvedl, jaké typy pravidel inline mód zavádí. Jedná se o následující tři.

drop – tento typ pravidla říká iptables, že má zahodit paket a uložit zprávu (log)

reject – při použití reject říkáme iptables, aby zahodil daný paket a uložil o tom zprávu, stejně jako u typu drop, navíc ale pošle TCP reset, v případě TCP, nebo ICMP port unreachable v případě UDP.

sdrop – pravidlo sdrop říká iptables, aby se paket zahodil, ale nezaznamenával tuto informaci do logu

kromě těchto třech pravidel máme ještě čtvrtou možnost a to nahradit určitý řetězec dat jiným řetězcem, musí ovšem platit, že počet nahrazených symbolů je stejný jako počet nahrazovaných symbolů. Ukázku nahrazení řetězce a tedy pozměnění obsahu paketu vidíte na následujícím výpisu (Výpis 3.3.1)

```
alert udp any any <> any 53 (msg: "udp nahrazeni"; \
        content: "google"; replace: "seznam";)
```

Výpis 3.3.1 Příkaz pro úpravu obsahu paketu

3.4. Režim síťového detektoru narušení (IDS)

Je zřejmé, že režim sledování provozu a režim ukládání systémových zpráv zachycují velké množství dat, což není pro co nejrychlejší detekci narušení zrovna ideální. A to z toho důvodu, že se musí toto velké množství dat dále analyzovat. Právě režim síťového detektoru narušení přináší řešení, jak detekovat podezřelý provoz a nestřádat přitom velký objem zachycených paketů. Tento režim je totiž řízen pomocí pravidel, jež definuje přímo výrobce detekčního programu nebo si je může definovat sám administrátor a přizpůsobit si je tak na vlastní síť. O těchto pravidlech se dozvíte více v následující kapitole. Soubor, který se na tyto pravidla odkazuje je snort.conf. Jestliže chci spustit program v režimu síťového detektoru, použiji následující příkaz (Výpis 3.4.1), který vám následně popíšu.

```
snort -A console -c /etc/snort/snort.conf
```

Výpis 3.4.1 Spuštění SNORTu v režimu síťového detektoru narušení

Při spuštění jsem použil několik přepínačů, jako první se zde vyskytuje `-A console` pro oznamovací (alert) mód s výpisem na obrazovku (console), přehled dalších mnou používaných přepínačů uvádím v tabulce níže (Tab. 3.4.1), více pak najdete v literatuře [9]. Následuje přepínač `-c`, jež SNORTu přikazuje, aby se řídil pravidly, ke kterým se dostane přes soubor snort.conf. Cesta k tomuto souboru tedy uzavírá zadaný příkaz.

Volba přepínače	Popis
`-A console`	Posílá hlášení o podezřelém provozu téměř v reálném čase na obrazovku.
`-A fast`	Mód rychlého upozornění, zapisuje hlášení se základními parametry provozu jako je časová známka, varovná zpráva, zdrojová a cílová IP adresa včetně portů.
`-A full`	Mód úplného upozorňování, je to základní nastavení, tento mód se použije v případě, že nespecifikujete konkrétní režim
`-A none`	Vypnutí oznamování.

Tab. 3.4.1 Možností přepínačů spuštění SNORTu v režimu síťového detektoru narušení

Jak již jsem zmiňoval, odkazy na soubory pravidel jsou uvedeny v konfiguračním souboru snort.conf. Po instalaci SNORTu je již celá řada pravidel vytvořena a připravena k použití. Pravidla jsou většinou rozdělena podle svého účelu či protokolu a jako příponu používají *.rules*. V případě OS Linux se jedná spíše o formalitu, jelikož tento OS nerozlišuje koncovky souborů, jako je tomu například u MS Windows. Cesta k základním pravidlům v adresářové struktuře je následující */etc/snort/rules/*, ale uživatel si pro své vlastní pravidla může zvolit libovolné umístění. Já si následně vytvořil vlastní soubor pravidel s názvem uzivatelska_pravidla.rules a zablokoval jsem základní pravidla vytvořena při instalaci SNORTu. Jednal jsem tak z toho důvodu, abych měl jistotu, že vyvolané varovné zprávy vychází z mých pravidel a nikoliv z těch základních. V praxi bych ovšem zablokování všech předdefinovaných pravidel určitě nedoporučoval, jelikož autoři programu mají aktuální informace o nových hrozbách ze stran útočníků a proto vytváří aktualizace

pravidel. Jedině tak dokáže IDS systém včas zareagovat na případné vniknutí útočníka do naší sítě. Při ukládání zachycených paketů pak SNORT standardně využívá složku /var/log/snort pokud mu nedefinujeme vlastní umístění. Nyní se už můžeme dostat k další kapitole, ve které se dozvíte o vytváření vlastních pravidel více.

4. Návrh a praktická implementace IDS systému

4.1. Základy

V této kapitole uvádím řadu vzorových pravidel pro řízení SNORTu v IDS režimu. Je proto dobré uvědomit si, že jsou umístněna v souboru pravidel. Využíval jsem soubor umístněný v adresářové struktuře /etc/snort/rules/uzivatelska_pravidla.rules. V předchozí kapitole jsem rovněž uvedl příkaz pro spuštění SNORTu v IDS režimu, pro připomenutí jej uvádím níže (Výpis 4.1.1). Popis tohoto výpisu naleznete v kapitole 3.4.

```
snort -A console -c /etc/snort/snort.conf
```

Výpis 4.1.1 Spuštění SNORTu v režimu síťového detektoru narušení

Kdybyste si nebyli jisti, co do příkazu zadat a v jakém formátu, máte možnost využít nápovědy použitím otazníku. Pro začátek si vyzkoušejte, co vyvolá příkaz `snort ?`. Zde ovšem předpokládám, že již máte SNORT nainstalován na svém počítači. Já k tomu využil příkaz `apt-get install snort`. Při instalaci a spouštění programu jsem býval přihlášen jako správce počítače - root. Soubory pravidel můžete upravovat i během chodu programu. Provedené změny se ale projeví až po jeho restartování. K tomu je vhodné vyvolat přerušení pomocí klávesové zkratky ctrl+c a opakovat příkaz z výpisu výše (Výpis 4.1.1). Může se také stát, že k přerušení SNORTu dojde až ve chvíli kdy, zachytí další paket v síti.

Psaní pravidel využívá syntaxe jednoduché, ale zároveň velmi flexibilní a efektivní. Každé pravidlo má jeden řádek. SNORT neumí pracovat s víceřádkovým zápisem, výjimkou je použití lomítka „\‘‘. To ovšem slouží spíše pro zpřehlednění daného pravidla.

Pravidlo ve SNORTu je pomyslně rozděleno na dvě části - záhlaví pravidla a parametry pravidla. Záhlaví obsahuje akci, protokol, zdrojovou a cílovou IP adresu včetně masky a také zdrojový a cílový port. Druhá část, tedy parametry pravidla, pak obsahují zprávu výstrahy a upřesnění částí paketů, které bude SNORT zkoumat.

Takto by pak mohlo vypadat jednoduché pravidlo ve SNORTu (Výpis 4.1.1).

```
alert icmp 10.0.2.0/24 any -> any any (msg: "ICMP zprava"; \
                    itype: 8; icode: 00;)
```

Výpis 4.1.1 Ukázka pravidla Snortu

Text před závorkami je záhlaví pravidla, uvnitř závorek jsou pak parametry pravidla. Slovo před každou dvojtečkou se nazývá klíčové slovo (neboli keyword). Zde není přesně určeno, kolik těchto klíčových slov v pravidle mohu použít, ale čím víc jich bude pravidlo obsahovat, tím více specifikuji paket na který SNORT zareaguje. To znamená, že aby SNORT zareagoval, tak musí procházející paket splňovat všechny parametry v pravidle. Dalo by se říct, že parametry uvnitř jednoho pravidla pojí logická funkce AND. Pravidel však může být více a mají mezi sebou vztah OR. Teď bychom se mohli podívat blíže na záhlaví pravidla.

4.2. Záhlaví pravidla

Akce pravidla:

Zde v akci pravidla řeknu SNORTu, co se má stát ve chvíli, kdy procházející paket splňuje všechny prvky z parametrů pravidla. Na výběr mám tři možnosti a jsou to alert, log a pass. Alert vypíše na obrazovce varovnou zprávu a uloží log o paketu. Akce log nebude na obrazovce nic vypisovat, ale uloží pouze systémovou zprávu o podezřelém provozu. Poslední možnost je pass a ta bude daný paket ignorovat a propustí jej. Akce pass se pak využívá spíše u IPS systémů.

Protokol:

Za určením akce pravidla následuje protokol a opět máme na výběr ze tří možností. Jsou to TCP, UDP nebo ICMP. Ve verzi SNORTu, kterou jsem používal (verze 2.8.4.1 – build 38) jsou to opravdu pouze tyto tři, ale autoři programu slibují, že do budoucna bude možný výběr z protokolů ARP, IGRP, GRE, OSPF, RIP, IPX a dalších.

IP adresa:

Třetí částí záhlaví pravidla jsou IP adresy včetně masek a portů, definovaných pro zdroj a cíl. Zde mám také možnost využít zkratky „any“ což už z překladu znamená „jakákoliv“ adresa. Adresa se zapisuje klasicky po čtyřech osmibitových číslech v desítkové soustavě oddělené tečkami. Za takto zapsanou adresou následuje CIDR blok. CIDR blok /8 potom označuje adresy z třídy A, blok /16 adresy z třídy B a blok /24 jsou adresy z třídy C. Příkladem adresy včetně CIDR bloku by mohlo být 172.16.0.0/16 a takový zápis tedy znamená, že máme k dispozici rozsah adres od 172.16.0.0 do 172.31.255.255. Dalo by se říct, že nám použití CIDR bloku značně ulehčuje zápis, kdybych zde musel vypisovat celou masku, bylo by každé pravidlo zbytečně dlouhé. Na

prvním výpise této kapitoly (Výpis 4.1.1), pravidlo reaguje na pakety jdoucí z rozsahu adres 10.0.2.0 s maskou /24 a cílová adresa je nastavena na „any“ tedy na všechny cílové adresy.

Mohu také využít symbol negace, což je u SNORTu vykřičník „!“ a pomocí něj pravidlo nastavím tak, aby si všímal všech ostatních adres kromě té zadané. Na následujícím výpise (Výpis 4.2.1) je tedy modifikované pravidlo z předchozí ukázky (Výpis 4.1.1).

```
alert icmp !192.168.1.0/24 any -> 192.168.1.0/24 any (msg: "ICMP zprava"; \
                                itype: 8; icode: 00;)
```

Výpis 4.2.1 Pravidlo s negací IP adresy

Takovým pravidlem pak mohu jednoduše říct, aby SNORT reagoval na všechny pakety, které jdou z vnější sítě do naší vnitřní. Neboli na pakety, které přichází ze všech rozsahů adres, kromě toho vnitřního.

Čísla portů:

Podobně jako u zápisu IP adresy, kde jsem měl více možností zápisu (s negací, případně pomocí slova any), tak i při určení zdrojového a cílového portu mohu využít různých variant. Použitím „any“ bude pravidlo reagovat na veškeré porty. Jestliže chci sledovat jeden konkrétní port, zadám jeho hodnotu. Například 23 pro telnet, 25 pro SMTP, 80 pro HTTP, nebo 220 pro IMAP atd. Můžu definovat také rozsah portů, v tom případě do zápisu přidám symbol dvojtečky „:“. Zápis pak bude vypadat následovně (Výpis 3.2.2).

```
alert tcp any any -> any 1024:2048 (msg: "TCP zprava"; ack: 10; seq: 15;)
```

Výpis 4.2.2 Pravidlo s určením rozsahu cílových portů

Pokud pravidlo mírně upravím a zapíši jej jako na výpisu 4.2.3 (Výpis 4.2.3), nastavím rozsah sledovaných cílových portů na 1024 (včetně) a výš, bez omezení nejvyššího portu.

```
alert tcp any any -> any 1024: (msg: "TCP zprava"; ack: 10; seq: 15;)
```

Výpis 4.2.3 Pravidlo s určením rozsahu cílových portů, více než

Analogicky mohu tento zápis otočit a nastavit rozsah cílových portů na 1024 (včetně) a všechny nižší (Výpis 4.2.4).

```
alert tcp any any -> any :1024 (msg: "TCP zprava"; ack: 10; seq: 15;)
```

Výpis 4.2.4 Pravidlo s určením rozsahu cílových portů, méně než

Poslední možností jak nadefinovat sledované porty je pomocí negace. V případě že před portem, nebo rozsahem portů použiji symbol vykřičníku „!“ říkám, aby SNORT sledoval všechny ostatní porty, kromě těch zadaných. Na výpisu 4.2.5 (Výpis 4.2.5) je ukázka pravidla, které sleduje cílové porty 0 až 499 a 799 až 65535.

```
alert tcp any any -> any !500:800 (msg: "TCP zprava"; ack: 10; seq: 15;)
```

Výpis 4.2.5 Pravidlo s určením rozsahu cílových portů s negací

Směr:

Posledním prvkem, který se vyskytuje v záhlaví pravidla SNORTu je symbol směru. Při tvorbě této práce jsem nejdříve definoval zdrojový rozsah adres a portů a následně cílové adresy a porty. Směr jsem pak definoval symbolem „->“ obecný zápis je na výpise níže (Výpis 4.2.6)

```
Zdrojová adresa port -> Cílová adresa port
```

Výpis 4.2.6 Pravidlo s určením rozsahu cílových portů s negací

Otočením můžeme zdrojovou a cílovou adresu prohodit, symbol by pak vypadal následovně „<-“ případně zápis pro obousměrnou reakci je tento „<>“.

4.3. Parametry pravidla

V parametrech pravidla se teprve odehrává to nejdůležitější. Je to v podstatě jádro, na kterém je systém detekce postaven. Právě v této části pravidla můžeme zpřísnit podmínky paketů, na které bude SNORT reagovat. Počet těchto parametrů v jednom pravidle není nijak omezen. Každý parametr má své klíčové slovo, za ním následuje dvojtečka a hodnota parametru. Parametry jsou mezi sebou odděleny symbolem středníku „;“. Zde je několik základních klíčových slov, které jsem ve SNORTu používal.

msg – zpráva, která se vypisuje na obrazovku

logto – soubor, do kterého se uloží log, namísto defaultně nastaveného souboru

minfrag – definice nejmenšího možného fragmentování, které bude akceptováno

ttl – definice hodnoty ttl

id – definice hodnoty ID pole v záhlaví paketu

content – konkrétní obsah zprávy

dsize – kontrola velikosti užitečné informace přenášeného paketu

offset – upřesnění slova content a určení kolik znaků přeskočí, než začne sledovat shodu

depht – upřesnění slova content, určení kolik znaků po sobě se má shodovat

flags – kontrola příznaku v TCP paketech

seq – kontrola sekvenčního čísla (sequence) v TCP paketech

ack – test potvrzujícího čísla (acknowledge) v TCP paketech

itype – kontrola typu ICMP zprávy

icode – kontrola kódu ICMP zprávy

nocase – vypnutí rozlišování velkých a malých písmen

sid – identifikační číslo pravidla ve SNORTu

msg:

Za tímto klíčovým slovem bude následovat text, jenž se vypíše na obrazovku v případě, že došlo ke shodě všech parametrů pravidla. V případě, že bych chtěl vypsat znak, který má v syntaxi pravidla jiný význam, například uvozovky (začátek a konec vypisované zprávy), nebo středník (ukončení parametru pravidla), pak musím před tyto symboly zadat znak lomítka „/“. Obecná syntaxe zápisu je `msg: "text zprávy";` a příklad v konkrétním pravidle je uveden na výpisu 4.3.1 (Výpis 4.3.1).

```
alert tcp any any -> any any (msg: "TCP zprava"; ack: 10; seq: 15;)
```

Výpis 4.3.1 Nastavení klíčového slova msg

logto:

Klíčové slovo logto slouží k tomu, abych mohl ukládat zprávy o podezřelých paketech do uživatelem definovaného logovacího souboru. Pokud toto nebudeme definovat, SNORT automaticky ukládá logy do souboru var/log/snort. Syntaxe je následující `logto: "/etc/snort/logovana";`.

minfrag:

Jak už zkratka napovídá, jedná se o minimální fragment. Je-li paket rozdělen na dílčí fragmenty, můžeme v pravidle nastavit jejich minimální velikost. Užitečné to může být v případě, kdyby útočník chtěl posílat velmi malé pakety, aby se vyhnul detekčnímu mechanismu. Standardně se velikost fragmentů pohybuje od 256 bytů výše, proto je rozumné vytvořit si pravidlo reagující na pakety s menší fragmentací, než je právě tato. Ukázka je na výpisu 4.3.2 (Výpis 4.3.2)

```
alert tcp any any -> any any (minfrag: 256; msg: "podezřele malý fragment";)
```

Obr. 4.3.2 Nastavení klíčového slova minfrag

ttl:

Klíčové slovo ttl určuje specifickou dobu života paketů (time to live). SNORT toto pravidlo obsahuje kvůli sledování pokusů o trasování paketů (traceroute). Syntaxe zápisu je `ttl: "číselný údaj";`.

id:

ID neboli identifikace fragmentu. Jedná se o 16 bitovou hodnotu, tedy s rozsahem 0 až 65535. Některé aplikace koncipované na narušení bezpečnosti sítí využívají známé identifikační čísla, a proto můžeme tyto id sledovat. Syntaxe je podobná jako u ttl, tedy `id: "číselný údaj";`.

dsize:

Tento parametr lze uplatnit při testování velikosti užitečné informace. Můžeme zde zadat libovolné číslo a využít symbolů větší než „>“ a menší než „<“, pro udání rozsahu velikostí. Syntaxe je obdobná jako v předchozích případech, ale tentokrát číslo není v uvozovkách. Příklad pravidla, které zahrnuje poslední tři zmíněné parametry, je na výpise 4.3.3 (Výpis 4.3.3).

```
alert tcp any any -> any any (ttl: "128"; id: "27536"; dsize: 23; \
                              msg: "TCP zprava";)
```

Výpis 4.3.3 Nastavení ttl, id a dsize

Na tomto výpise (Výpis 4.3.3) je dobře vidět, jak můžeme parametry pravidla kombinovat.

content:

Klíčové slovo content mě překvapilo, pomocí něj totiž uživatel SNORTu může nechat prohledávat užitečné informace v přicházejících datech. V pravidle pak uvádím řetězec znaků, které se mají v datech vyskytnout a v případě shody může SNORT reagovat. Zde je vhodné uvést skutečnost, že SNORT rozlišuje malá a velká písmena. Řetězec nemusí být zadaný pouze jako text (Výpis 4.3.4), ale i jako binární byte kódu (Výpis 4.3.5). Binární byte kód reprezentuje binární hodnoty jako hexadecimální čísla. Binární byte kód se musí uzavřít mezi roury neboli pipes „|“. Oba typy zápisu pak mohu zkombinovat (Výpis 4.3.6).

```
alert tcp any any -> any any (content: "virus"; \
        msg: "Obsahuje slovo virus";)
```

Výpis 4.3.4 Definování content jako textu

```
alert tcp any any -> any any (content: "|76 69 72 75 73|"; \
        msg: "Obsahuje slovo virus";)
```

Výpis 4.3.5 Definování content binárním byte kódem

```
alert tcp any any -> any any (content: "|6E 6F 76 79| virus"; \
        msg: "Obsahuje slovo virus";)
```

Výpis 4.3.6 Definování content kombinací textu a binárního byte kódu

offset:

Zde se jedná o klíčové slovo, které se vyskytuje v pravidlech obsahující parametr content. Znamená to, že parametr content je podmínkou pro použití offsetu. Pomocí něj mohu posunout prohledávání o příslušný počet znaků. S tímto parametrem by se mělo zacházet opatrně, jelikož by mohl zabránit reakci na skutečný útok. Syntaxe je `offset: číselný výraz;`. Ukázka použití je uvedena na výpise 4.3.7 (Výpis 4.3.7) společně s parametrem depth.

depth:

Stejně tak jak tomu bylo u parametru offset, i u depth je pro jeho použití podmínka, aby pravidlo obsahovalo parametr content. Depth neboli hloubka označuje, kolik znaků v užitečné informaci se má maximálně prohledávat. Syntaxe je `depth: číselný výraz;`.

```
alert tcp any any -> any any (content: "virus"; offset: 15; depth:150; \
                    msg: "obsahuje text virus";)
```

Výpis 4.3.7 Nastavení parametrů offset a depth

nocase:

Jak už jsem se zmiňoval v této práci, SNORT rozlišuje velká a malá písmena, což u pravidel reagujících na určitý řetězec znaků, tedy těch, jež obsahují klíčové slovo content, může znamenat problém. Například u přenosu exe souboru z FTP serveru, nemusí být přípona souboru vždy napsaná malými písmeny. Abych pak nemusel psát mnoho pravidel, pro různé kombinace velkých a malých písmen, mohu využít právě klíčového slova nocase. Syntaxe je jednoduchá, stačí toto klíčové slovo zadat mezi parametry pravidla a ukončit středníkem. Ukázka použití je na výpise 4.3.8 (Výpis 4.3.8).

```
alert tcp any any -> any 20 (nocase; content: "exe"; \
    msg: "prenos spustitelného souboru přes FTP";)
```

Výpis 4.3.8 Pravidlo reagující na přenos exe souboru z FTP serveru

flags:

Tento parametr má své uplatnění při sledování TCP paketů a prohledává tedy příznak. Každý příznak má v tomto parametru své označení a to následující:

A – ACK – acknowledgement number
F – FIN – end of data
P – PSH – push
R – RST – reset connection
S – SYN – synchronize sequnce number
U – URG – urgent pointer

Syntaxe je následující `flags: příslušné označení;`. Ukázka je uvedena na výpise 4.3.9 (Výpis. 4.3.9)

seq:

Sequence number, je 32bitové sekvenční číslo, jež naleznete v hlavičce TCP zprávy. SNORT zjišťuje číselnou hodnotu sekvenčního čísla. Syntaxe je `seq: číselná hodnota;`. Ukázka je uvedena na výpise 4.3.9 (Výpis 4.3.9).

ack:

Podobně jako sekvenční číslo, tak i acknowledgement number, je 32bitové číslo v TCP hlavičce. V překladu se jedná o potvrzovací číslo a syntaxe zápisu v pravidle SNORTu je takováto `ack: číselná hodnota;`. Ukázka parametrů flags, seq a ack je na následujícím výpisu 4.3.9 (Výpis 4.3.9).

```
alert tcp any any -> any any (flags :AF; seq: 25; ack: 15; \
        msg: "podezřelý TCP paket";)
```

Výpis 4.3.9 Užití parametrů flags, seq a ack

itype:

Tento parametr se používá u sledování ICMP zpráv, takže i v záhlaví pravidla SNORTu se tentokrát nevyskytne alert tcp, ale alert icmp. Seznam příslušných typů lze nalézt na internetových stránkách organizace IANA viz literatura [14]. Syntaxe je následující `itype: číselná hodnota;`. A ukázka je na výpisu 4.3.10 (Výpis 4.3.10).

icode:

Dalším prvkem definujícím ICMP zprávu je kód. Opět je seznam kódů uveden na stránkách organizace IANA viz literatura [14]. Syntaxe je `icode: číselná hodnota;`. Ukázka je na výpisu 4.3.10 (Výpis 4.3.10).

```
alert icmp any any -> any any (itype: 08; icode: 00; msg: "ICMP Echo";)
```

Výpis 4.3.10 Nastavení pravidla na zprávu typu ICMP Echo

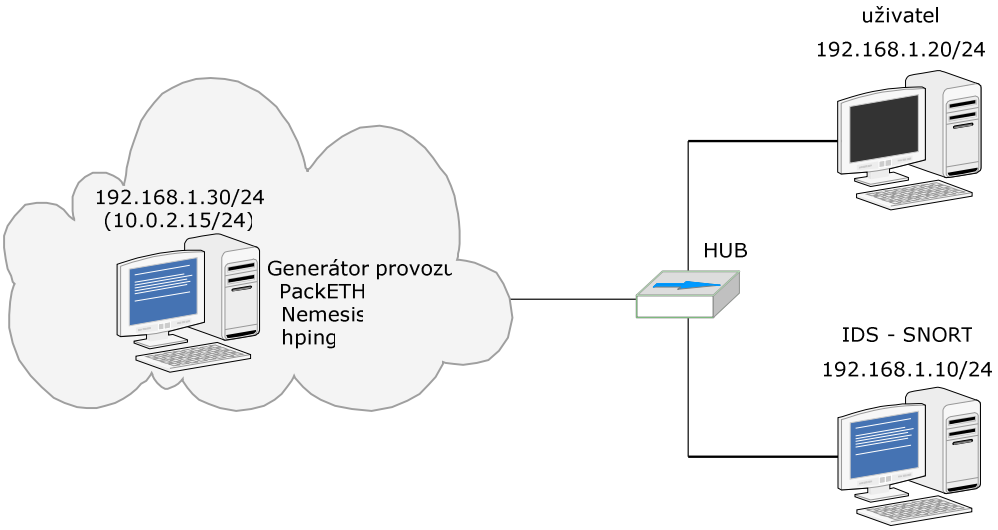
sid:

Jako poslední parametr uvádím SID pravidla. Zde se nejedná o parametr, který by ovlivnil prohledávání obsahu paketů, ale je to identifikační číslo daného pravidla. V případě, že tento parametr neuvedete, SNORT při spuštění zahlásí chybu. U zadávání SID platí tyto podmínky:

- < 100 - rezervováno pro využití v budoucnu
- 100 – 1.000.000 - pravidla obsažená ve SNORTu
- > 1.000.000 - využito pro lokální pravidla

4.4. Praktická ukázka pravidel a reakce IDS SNORT

Dříve jsem se zmínil, že mnou vytvořená pravidla jsem vepisoval do vlastního souboru pravidel uzivatelska_pravidla.rules. Tento soubor mám umístěn v adresářové struktuře /etc/snort/rules/uzivatelska_pravidla.rules. Schéma konkrétního zapojení, na kterém jsem IDS režim SNORTu testoval, uvádím na následujícím obrázku (Obr. 4.4.1). K upřesnění bych si dovolil vysvětlit, proč je u počítače s generátorem provozu uvedena adresa 10.0.2.15/24 v závorce. Je to proto, že jsem nevyužíval žádné směrovací protokoly, a tudíž jsem na síťové kartě nastavil adresu z lokální sítě (192.168.1.30), to mi zajistilo konektivitu s ostatními prvky v síti. V samotném penetračním nástroji, generuji pakety se zdrojovou IP adresou z vnější sítě, tedy s adresou 10.0.2.15/24.



Obr. 4.4.1 Schéma zapojení IDS

Jak SNORTu říci, aby se na vlastní soubor pravidel odkazoval, a jaký je jeho obsah, se dočtete v následujících odstavcích.

Nejprve jsem si vytvořil prázdný textový dokument s názvem a umístěním uvedeným v úvodu této podkapitoly. Abych jej zahrnul do souboru pravidel, podle kterého se bude SNORT řídit, musel jsem do konfiguračního souboru snort.conf uvést odkaz na tento soubor. Na následujícím výpisu (Výpis 4.4.1) je uveden příklad zadání odkazu.

```
include $RULE_PATH/uzivatelska_pravidla.rules
```

Výpis 4.4.1 Odkaz na vlastní soubor pravidel ve snort.conf

Zde si můžete všimnout, že se místo celé cesty k uživatelskému souboru pravidel, vyskytuje jakási zkrácená verze, tedy s využitím proměnné RULE_PATH, jež nám nahrazuje cestu /etc/snort/rules. V případě, že byste si vytvořená pravidla umístili do jiné složky, nesmíte zapomenout uvézt celou cestu k vámi vytvořenému souboru, nebo si definovat proměnnou pro vlastní umístnění. Takováto proměnná se definuje tak, jak je uvedeno na výpise níže (Výpis 4.4.2)

```
var RULE_PATH /etc/snort/rules
```

Výpis 4.4.2 Definování proměnné RULE_PATH ve snort.conf

Proměnné opět definujeme v konfiguračním souboru snort.conf. Ve chvíli kdy se budu chtít na proměnnou odkázat, musím před název proměnné RULE_PATH uvést symbolem dolaru „$“ jak je tomu na výpise 4.4.1 (Výpis 4.4.1). Využití proměnných nemusí být pouze u zjednodušení cesty k souboru pravidel. Jako příklad dalšího využití mohu uvést v souvislosti s definováním rozsahu adres v záhlaví pravidel. Mohu si tak nastavit proměnnou MOJE_sit, která bude nahrazovat rozsah adres v mé lokální síti, například 192.168.1.0/24. Výpis ze snort.conf je následující (Výpis 4.4.3).

```
var MOJE_sit 192.168.1.0/24
```

Výpis 4.4.3 Definování proměnné pro rozsah adres

Nyní už se konečně dostávám k samotnému obsahu mého souboru pravidel (Výpis 4.4.4).

```
# soubor vlastnich pravidel - Jiri Dona
# nazev a umisteni souboru: /etc/snort/rules/uzivatelska_pravidla.rules
# posledni zmena 12.4.2010

# 1. pravidlo pro oznameni ICMP paketu
alert icmp 10.0.2.0/24 any -> 192.168.1.0/24 any (itype: 08; icode: 00; \
msg: "ICMP Echo zprava"; sid:1250000;)

# 2. pravidlo pro oznameni podezreleho TCP paketu
alert tcp any 1024: -> 192.168.1.0/24 any (flags: A; ack: 20; seq: 21; \
msg: "TCP paket s flagem A"; sid:1260000;)

# 3. pravidlo pro oznameni podezreleho UDP paketu
alert udp 10.0.2.0/24 any -> any 53 (msg: "UDP paket na port pro DNS"; \
sid:1270000;)
```

```
# 4. pravidlo pro oznameni prenosu z FTP serveru
alert tcp 10.0.2.0/24 any -> 192.168.1.0/24 21 (msg: "FTP prenos"; \
content: ".exe"; sid:1280000; nocase;)

# 5. pravidlo pro oznameni podezreleho e-mailu
alert udp any 110 -> 192.168.1.0/24 (msg: "podezrely e-mail"; \
content: "posilam virus"; sid:1290000; nocase;)

#konec souboru
```

Výpis 4.4.4 Obsah vlastního souboru pravidel

Při tvorbě vlastních pravidel je vhodné uvádět poznámky, já uvedl v úvodu souboru jméno autora, původní název, umístění a datum poslední aktualizace souboru. Tyto poznámky musí mít na začátku řádku mřížku „#“ tím SNORT rozlišuje co je část konfigurace a co není. Abych tyto pravidla mohl otestovat, použil jsem softwarové penetrační nástroje, o kterých se dočtete v následující kapitole. Na výpisu níže (Výpis 4.4.5) je zachycena reakce SNORTu podle prvních třech pravidel uvedených na výpisu 4.4.4 (Výpis 4.4.4).

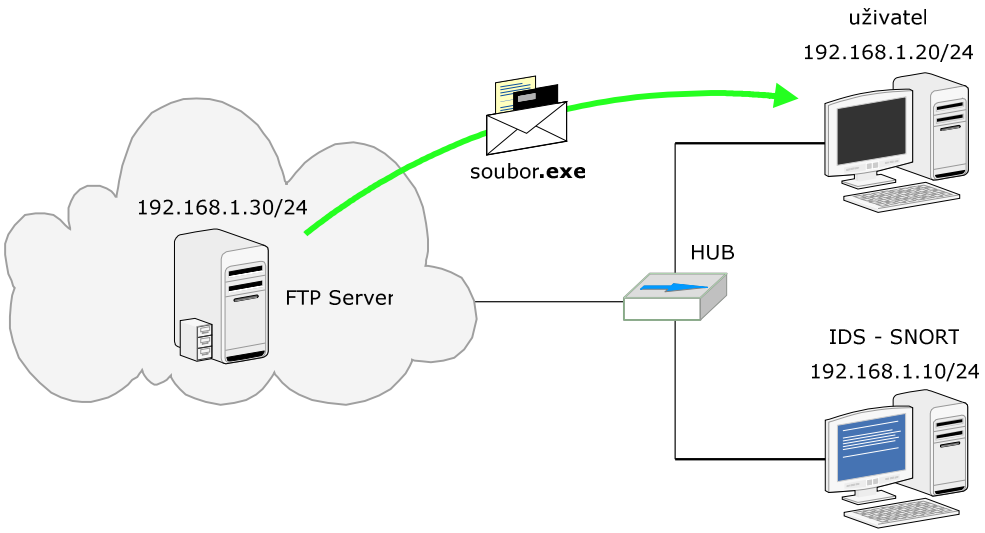
```
①root@student-desktop:~# snort -A console -c /etc/snort/snort.conf
Running in IDS mode

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.8.4.1 (Build 38)
   ''''    By Martin Roesch & The Snort Team: http://www.Snort.org/team.html
②          Copyright (C) 1998-2009 Sourcefire, Inc., et al.
           Using PCRE version: 7.8 2008-09-05

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.10  <Build 16>
Not Using PCAP_FRAMES
③04/22-10:23:03.153425  [**] [1:1250000:0] ICMP Echo zprava [**] [Priority: 0]
{ICMP} 10.0.2.15 -> 192.168.1.20
④04/22-10:23:05.154487  [**] [1:1260000:0] podezrely TCP paket s flagem A [**]
[Priority: 0] {TCP} 10.0.2.15:1111 -> 192.168.1.20:666
⑤04/22-10:23:07.154562  [**] [1:1270000:0] UDP paket na port pro DNS [**]
[Priority: 0] {UDP} 10.0.2.15:5966 -> 192.168.1.20:53
^C*** Caught Int-Signal
Run time prior to being shutdown was 30.786784 seconds
===============================================================================
```

```
Packet Wire Totals:
   Received:            3
⑥  Analyzed:            3 (100.000%)
    Dropped:            0 (0.000%)
Outstanding:            0 (0.000%)
===============================================================================
Breakdown by protocol (includes rebuilt packets):
      ETH: 3           (100.000%)
      IP4: 3           (100.000%)
⑦     TCP: 1           (33.333%)
      UDP: 1           (33.333%)
     ICMP: 1           (33.333%)
    Total: 3
===============================================================================
Action Stats:
ALERTS: 3
⑧ LOGGED: 3
PASSED: 0
===============================================================================
Snort exiting
```

Výpis 4.4.5 reakce na provoz podle vlastních pravidel

Ve výše uvedeném výpisu je pak vidět zadaný příkaz pro spuštění SNORTu v IDS režimu ①, dále pak informace o verzi a autorovi aplikace ②. Poté již následují samotné varovné zprávy. První tedy reaguje na ICMP zprávu ③, druhá na TCP zprávu ④ a poslední na UDP zprávu ⑤. Samozřejmě že by tyto zprávy SNORT nevyprodukoval v případě, že by generované pakety neměly takové parametry, jaké jsou nastaveny v pravidlech uvedené na výpisu 4.4.4 (Výpis 4.4.4). Jakmile SNORT zareagoval varovnými zprávami, vyvolal jsem přerušení programu (ctrl+c) a SNORT vygeneroval statistiku své činnosti. Pod číslem ⑥ je uveden přehled o počtu přijatých (received) a analyzovaných (analyzed) paketů. Pod číslem ⑦ jsou tyto počty vyjádřeny procentuálně a v části výpisu označené ⑧ jsou informace o počtech akcí, tedy počet oznámení (alerts) a počet uložených zpráv (logged).

Kromě těchto generátorů provozu jsem pro otestování pravidel využil i reálný provoz, jež jsem simuloval přenosem spustitelného souboru z FTP serveru a příjmem elektronické pošty obsahujícího text „posilam virus“. Pravidla pro tento provoz jsou uvedena jako poslední dvě v souboru mých vlastních pravidel (Výpis 4.4.4). Schéma zapojení pro FTP přenos uvádím na následujícím obrázku (Obr. 4.4.2)



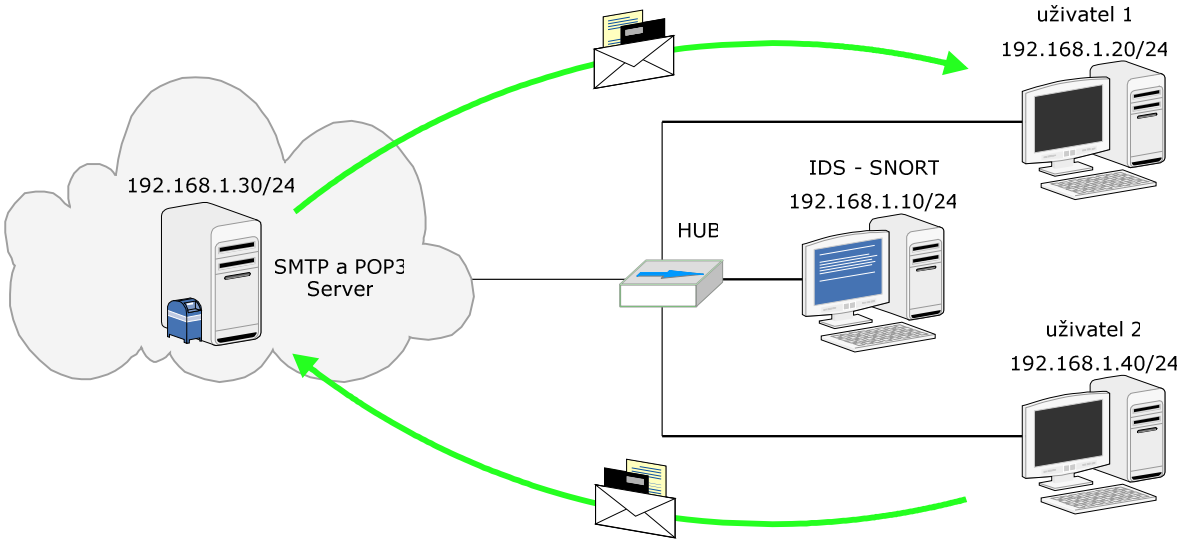
Obr. 4.4.2 Schéma zapojení FTP serveru

Oblak, ve kterém se nachází FTP server, reprezentuje vnější síť, ale pro zjednodušení jsem použil umístění tohoto serveru do vnitřní sítě, což je patrné podle přidělených IP adres. Ve chvíli, kdy se tedy uživatel pokusí přenést soubor.exe, zareaguje IDS SNORT varovnou zprávou. Reakce je na výpisu (Výpis 4.4.6). Z výpisu je také patrné, že soubor odcházel z IP adresy FTP serveru a portu 55597 a mířil na adresu uživatele, kde cílový port 21 upřesňuje skutečnost, že se jedná o FTP přenos.

```
04/22-19:09:25.305531  [**] [1:1280000:0] FTP prenos [**] [Priority: 0] {TCP}
192.168.1.30:55597 -> 192.168.1.20:21
```

Výpis 4.4.6 Varovná zpráva při detekci přenosu z FTP serveru

Druhým příkladem reakce na reálný provoz je oznámení podezřelého obsahu e-mailové zprávy. Pro tento případ jsem sestavil následující schéma (Obr. 4.4.3).



Obr. 4.4.3 Schéma zapojení SMTP a POP3 Serveru

Stejně jako u FTP Serveru, tak i zde jsem umístil poštovní server do vnitřní sítě. Jakmile uživatel 2 odešle e-mail, který obsahuje text „posilam virus“ IDS jej zachytí ve chvíli kdy tento e-mail odchází z poštovního serveru směrem k uživateli 1. Stane se tak proto, že pravidlo je nastaveno na zdrojový port 110 což je port pro POP3 neboli protokol pro příchozí poštu. Reakce SNORTu je vyobrazena na následujícím výpise (Výpis 4.4.7).

```
04/22-18:48:56.763497  [**] [1:1290000:0] podezrely e-mail [**] [Priority: 0]
{TCP} 192.168.1.30:110 -> 192.168.1.20:47743
```

Výpis 4.4.7 Varovná zpráva při detekci podezřelého e-mailu

Otestování IDS pravidel přímo na reálném provozu bych v praxi nedoporučil, jelikož v reálném provozu může sítí putovat spousta paketů, o jejichž obsahu můžeme jedině spekulovat. Proto je při testování vhodné použít generátorů provozu a využít tak možnosti vytvářet si pakety na míru. Vygenerovaný provoz má pak přesně ty parametry, na které má SNORT reagovat varovnou zprávou. V následující kapitole vám některé z těchto generátorů provozu neboli penetračních nástrojů představím.

5. Nástroje pro ověření funkčnosti IDS systému

Penetračních nástrojů neboli generátorů síťového provozu, jsem měl při psaní této práce k dispozici celou řadu. Mohl jsem vybírat z placených aplikací nebo z volně dostupných či dokonce zdrojově otevřených. Nakonec jsem si pro otestování pravidel řídících IDS systém SNORT vybral tyto tři, packETH, Nemesis a hping.

5.1. packETH

PackETH je paketový generátor s grafickým rozhraním pro Linux a od verze 1.6 i pro Windows. Umožnil mi vytvářet a rozesílat jednotlivé pakety, nebo sekvenci paketů. Od letošního roku, tedy 2010 a verze programu 1.6, packETH podporuje kromě IPv4 i možnost vytvářet pakety směrované do sítí s IPv6. Vývoj programu však stále trvá, takže podpora IPv6 není zcela dokončena. Může se vám stát, že některé tlačítka a funkce packETHu nebudou při použití IPv6 fungovat. V aktuální verzi tedy můžete v IPv6 vytvářet a rozesílat TCP a UDP pakety. Já ovšem využíval adresování pomocí IPv4 a v tomto případě pracovaly všechny prvky packETHu bezproblémově. Konkrétní vlastnosti packETHu jsou tedy následující:

Podporované protokoly

- ethernet II, ethernet 802.3, 802.1q, QinQ
- ARP, IPv4, IPv6, uživatelem definovaný obsah užitečné zprávy v síťové vrstvě
- UDP, TCP, ICMP, IGMP, uživatelem definovaný obsah užitečné zprávy v transportní vrstvě

Rozesílání sekvence paketů

- Zpoždění mezi pakety, počet paketů k odeslání
- Změna parametrů během rozesílání paketů (změna IP a MAC adresy, portů atp.)

Možnost ukládání konfigurací paketů a následné čtení těchto konfiguračních souborů

Nyní vám představím zmíněné grafické prostředí programu packETH. Jeho obrazovka se dělí do více záložek, jejichž obsah naleznete v příloze č.1. Pod první záložkou, pojmenované „builder“, nastavuji protokol (ethernet, IP, UDP, TCP, ICMP, …) dále zdrojovou a cílovou MAC adresu, zdrojovou a cílovou IP adresu a další parametry podle zvoleného protokolu. Například v případě UDP, nebo TCP určuji zdrojový a cílový port. Pokud vyberu TCP tak dále určím sekvenční číslo, potvrzující číslo, příznak atd. V případě ICMP protokolu definuji například typ a

kód zprávy. Pod druhou záložkou s názvem Gen-b, se naskýtá možnost upravit počet odeslaných paketů, jejichž parametry jsme nastavili v záložce builder. Dále časový rozestup mezi odesílanými pakety, a je možno upravit obsah užitečné zprávy. Třetí záložka, autory programu pojmenovaná jako Gen-s, umožňuje využití uložených nastavení paketů ze záložky builder. V záložce Gen-b jsem si mohl odeslat několik paketů za sebou s přesně definovaným časovým rozestupem, jednalo se však o totožné pakety, případně pozměněné v době mezi odesláním jednotlivých paketů. V záložce Gen-s ovšem mohu rozesílat pakety s různými parametry, které jsem si dopředu definoval a uložil. Jako příklad nahlédněte do přílohy č.1, kde posílám TCP, UDP a ICMP pakety s dvou sekundovým rozestupem. Místo uložení jejich konfigurace je zde také uvedeno, tedy /etc/packeth/TCP (UDP, ICMP). Pro odeslání paketů do sítě, je nutné zvolit rozhraní, přes které se paket odešle. Rozhraní jsem vybíral pomocí tlačítka „interface“.

Snímky uvedené v příloze č.1, odpovídají nastavení, které jsem použil při generování provozu na otestování vlastního souboru pravidel uzivatelska_pravidla.rules v aplikaci SNORT. Výpis SNORTu (Výpis 4.4.5) je tedy reakcí na pakety odpovídající nastavení v příloze č.1.

5.2. Nemesis

Nemesis je zdrojově otevřený, síťový generátor provozu, ale na rozdíl od PackETHu nedisponuje grafickým rozhraním. Je tedy řízen pomocí pokynů z příkazové řádky. Podpora Nemesisu je jak pro OS UNIX, tak pro MS Windows. Jeho účel je přímo pro testování IDS systémů, firewallů a dalších síťových prvků. Díky ovládání z příkazové řádky je tento program vhodný pro automatizaci a ovládání pomocí vytvořených skriptů.

Podporované protokoly

- UDP, TCP, ICMP, IGMP
- ARP, DNS, Ethernet, IP, OSPF, RIP

Podporované OS

- Linux, BSD, Solaris, Mac OS X, MS Windows

Požadavky na OS UNIX

- libnet-1.0.2a
- C compiler (GCC)

Požadavky na OS Windows

- libnetNT-1.0.2g
- WinPcap-2.3 or WinPcap-3.0

Základní informace o Nemesisu tedy máte. Nyní bych vám ukázal jak tento síťový generátor použít. Na úvod je dobré znát syntaxi zadávání příkazů. Ta nejzákladnější je uvedena na výpisu níže (Výpis 5.2.1).

```
nemesis [protokol][-klíčové slovo] [hodnota]
```

Výpis 5.2.1 Základní syntaxe síťového generátoru Nemesis

Při tvorbě této práce jsem využíval Nemesis pro generování TCP, UDP a ICMP paketů, v případě generování paketů ostatních typů protokolů, jako jsou OSPF, RIP, IGMP atd. je syntaxe opět stejná a parametry se budou lišit podle obsahu hlaviček jednotlivých protokolů. Více informací pak naleznete také v literatuře [11]. Nyní tedy konkrétní ukázka. Na následujícím výpise (Výpis 5.2.2) je příkaz pro vygenerování ICMP zprávy a pod ním je výstup Nemesisu, který mě informuje o parametrech vygenerovaného paketu.

```
root@student-desktop:~# nemesis icmp -d eth0 -S 10.0.2.15 -D 192.168.1.20
-M 00:40:F4:92:E9:91 -H 08:00:27:6A:CA:FB -i 8 -c 00 -qE -v

ICMP Packet Injection -=- The NEMESIS Project Vision 1.4beta3 (Build 22)

             [MAC] 00:40:F4:92:E9:91 > 08:00:27:6A:CA:FB
   [Ethernet type] IP (0x0800)

              [IP] 10.0.2.15 > 192.168.1.20
           [IP ID] 12199
        [IP Proto]
          [IP TTL]
          [IP TOS] 00
  [IP Frag offset] 0000
   [IP Frag flags]

       [ICMP Type] Echo Request
       [ICMP Code] Echo Request
         [ICMP ID] 29940
 [ICMP Seq number] 30850

Wrote 42 bzte ICMP packet through linktzpe DLT_EN10MB.

ICMP Packet Injected
```

Výpis 5.2.2 Generování ICMP paketu pomocí aplikace Nemesis

V předchozí podkapitole jsem penetračním nástrojem PackETH vytvořil paket na nějž SNORT zareagoval výpisem v kapitole 4.4 (Výpis 4.4.5). Při použití příkazu výše, by SNORT vypsal stejnou zprávu jako ve třetím bodě ③, dříve zmíněného výpisu (Výpis 4.4.5).

Příkazy pro vygenerování TCP a UDP paketu jsou následující (Výpis 5.2.3).

```
root@student-desktop:~# nemesis tcp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M
00:40:F4:92:E9:91 -H 08:00:27:6A:CA:FB -x 1111 -y 666 -s 21 -a 20 -fA -w 4000 -v

root@student-desktop:~# nemesis udp -d eth0 -S 10.0.2.15 -D 192.168.1.20
-M 00:40:F4:92:E9:91 -H 08:00:27:6A:CA:FB -x 5966 -y 53 -v
```

Výpis 5.2.3 Generování TCP a UDP paketu pomocí aplikace Nemesis

Popis zde uváděné konfigurace TCP, UDP a ICMP paketů uvádím v příloze č.2. Více informací, včetně nastavení dalších protokolů, pak naleznete v literatuře [11].

Mohlo by se zdát, že nevýhodou oproti PackETHu je nemožnost uložení konfigurace paketu a následná obnova ze souboru. Nemesis to ovšem kompenzuje využitím shall souborů. Abych docílil stejné reakce SNORTu jako u vygenerování ICMP, TCP a UDP paketu v aplikaci PackETH (Výpis 4.4.5), vytvořil jsem si vlastní shall soubor pojmenovaný mujshall.sh, jehož výpis uvádím v příloze č.2. Po spuštění tohoto shall souboru se vygenerují všechny tři pakety s příslušnými parametry. V případě potřeby bych mohl jednoduchou úpravou nastavit například časové rozestupy mezi pakety, či nastavit počet opakování, případně po každém vygenerování přičíst číslo k hodnotě cílového portu atp.

5.3. hping

Hping je dalším nástrojem pro generování paketů, ovládaným z příkazové řádky. Stejně jako u předchozích dvou generátorů mohu i pomocí hpingu nastavovat obsah hlaviček paketů TCP, UDP a ICMP protokolů. Operačních systémů, na kterých mohu hping nainstalovat a použít je celá řada, konkrétně jsou to Linux, FreeBSD, NetBSD, OpenBSD, Solaris, MacOs X a MS Windows. Tento software je přímo určen pro testování zabezpečení počítačových sítí. Pro jeho používání je vhodné znát syntaxi zápisu příkazů (Výpis 5.3.1).

```
hping [-klíčové slovo] [hodnota]
```

Výpis 5.3.1 Obecný zápis příkazu pomocí aplikace hping

Přehled vybraných klíčových slov uvádím v příloze č.3 a více o konfiguraci hpingu naleznete také v literatuře [8]. Na následujícím výpisu (Výpis 5.3.2) je uveden příklad příkazu, jehož výstupem bude stejný paket jako při použití nástroje Nemesis (Výpis 5.2.2) a (Výpis 5.2.3).

```
root@student-desktop:~# hping -I eth0 192.168.1.20 -s 1111 -p 666 -M 21 -L 20 -A
root@student-desktop:~# hping -2 -I eth0 192.168.1.20 -s 5966 -p 53
```

Výpis 5.3.2 Generování TCP a UDP paketu pomocí aplikace hping

Nevýhoda oproti PackETHu a Nemesisu, kterou v případě hpingu shledávám, je ta, že nemůžu navolit libovolnou zdrojovou IP adresu bez ohledu na tom, jaká IP adresa je nastavena na rozhraní, přes nějž paket vysílám. Hping neumožňuje ani nastavovat zdrojovou a cílovou MAC adresu. Jak již jsem zmiňoval, tento generátor je určen pro testování bezpečnosti sítí, proto disponuje funkcí port sken. Tedy zvyšování čísla cílového portu s každým odeslaným paketem. Odpovědí na tyto pakety je informace, jestli je port otevřen či nikoliv. Princip je založen na 3 way handshake, viz. příloha č.4. Příkaz pro port sken je uveden na následujícím výpisu (Výpis 5.3.3).

```
root@student-desktop:~# hping -I eth0 192.168.1.20 -c 3 -S ++p 666
```

Výpis 5.3.3 Příkaz pro automaticky se zvyšující číslo portu

Jak je z výpisu patrné, konfigurace postupného navyšování čísla portů se provádí pomocí dvou symbolů „++“ před klíčovým slovem cílového portu „p“. Odpovědí na požadavek navázání komunikace bude buď zpráva SYN-ACK a port je tedy otevřen, nebo zpráva RST-ACK, tedy odmítnutí navázání komunikace a port je uzavřen.

6. Závěr

Cílem mé diplomové práce bylo popsat teoretický úvod do problematiky detekce a prevence narušení počítačových sítí, dále navrhnout a v praxi implementovat vhodný softwarový nástroj, pro detekci narušení, aplikaci SNORT a nakonec ověřit funkčnost detekčního systému pomocí různých penetračních nástrojů. Veškeré konfigurační soubory, které jsem při tvorbě této práce použil, jsou uvedeny v příloze č.5.

V této práci objasňuji způsoby zapojení IDS a IPS zařízení do síťové struktury, včetně ukázky konfigurace přepínače s využitím zrcadlení portů. U detekčních systémů uvádím různé principy rozhodování o případném útoku. Jsou to systémy založené na signaturách, na detekci anomálního chování a detekci odchylek od protokolových norem. Dále se zaměřuji na detekční systém založený na signaturách s názvem SNORT. Tomuto systému je věnována třetí a čtvrtá kapitola. Ve třetí kapitole uvádím popis všech režimů, ve kterých umí SNORT pracovat. Jedná se o tři základní režimy, a to režim sledování provozu, režim ukládání systémových zpráv a režim detekce síťového narušení (IDS). Čtvrtý rozšiřující režim inline (IPS), byl do aplikace přidán až od verze 2.3.0. V následující čtvrté kapitole popisuji syntaxi zápisu pravidel v aplikaci SNORT. Uvádím strukturu pravidel včetně jejich názorných ukázek, a také výpisy z detekčního systému po průchodu podezřelého provozu. Vytvořená pravidla jsem testoval pomocí paketových generátorů, navíc jsem funkčnost pravidel ověřil i v reálném provozu. Pro reálný provoz jsem nakonfiguroval poštovní server a klienty, díky nimž jsem mohl rozesílat v lokální síti e-mailové zprávy. Navíc jsem zprovoznil FTP server a přenášel z něj data směrem k uživatelům.

Co se týče předností a nedostatků jednotlivých penetračních nástrojů, jeví se mi jako nejvýhodnější aplikace Nemesis. A to z toho důvodu, že je tento program velmi přehledný, příkaz pro vygenerování paketu se vejde na jeden až tři řádky. Dostupná dokumentace je kvalitně zpracována a považuji také za velkou výhodu možnost využití bash shell pro ovládání tohoto programu. Generátor PackETH mě ohromil svým grafickým rozhraním a z tohoto důvodu jsem jej pro testování využil jako první. Výhodou GUI je to, že nemusím znát konkrétní obsah hlaviček paketů jednotlivých protokolů, nemusím znát syntaxi zápisu, pro vytvoření paketu, jako u Nemesisu. Stačí vyplnit prázdná políčka v okně programu, vybrat rozhraní, ze kterého budu vysílat a stiskem tlačítka odešlu paket. Navíc je zde výhoda v možnosti ukládání nastavení paketů do souboru. Když jsem následně odesílal tři různé, po sobě jdoucí pakety, zadal jsem cestu k uložené konfiguraci, nastavil čas mezi pakety a opět stiskl tlačítko pro odeslání. Z toho plyne, že nemusím znát ani syntaxi zápisu v bash shell. Postupem času jsem ovšem zjistil, že toto grafické rozhraní

není tak dobré, jak se na začátku zdálo. To hlavně z toho důvodu, že okno prostředí PackETHu je větší než rozlišení obrazovky a navíc se dělí do několika záložek. Proto jsem se v okně ztrácel a neměl přehled o tom, jaké parametry jsem paketu nastavil. Posledním generátorem provozu, který jsem zkoušel je hping. Konfiguraci paketů v této aplikaci jsem prováděl opět z příkazové řádky jako v případě Nemesisu, ovšem způsob zápisu příkazů u hpingu není tak pohodlný. Nemesis má přesně rozdělené protokoly jako jsou TCP, UDP, ICMP a další, proto je psaní příkazů velmi přehledné. Hping v syntaxi nahrazuje například název protokolů číslicí. Další nevýhodou hpingu je špatná dostupnost dokumentace a při testování jsem si proto musel vystačit pouze s nápovědou integrovanou v programu. Posledním nedostatkem, který jsem odhalil, bylo, že jsem nemohl nastavit zdrojovou a cílovou MAC adresu ani zdrojovou IP adresu jinou, než jsem měl nakonfigurovanou na rozhraní. Vlastností hpingu, kterou se odlišuje od konkurence je, že umožňuje nastavit automatické přičítání cílového portu a provádět tak port sken. U Nemesisu jsem to ale vyřešil proměnnou v bash shell a vložením příkazu do cyklu, ve kterém se proměnná navyšovala.

Při tvorbě této práce jsem vybíral aplikace z nabídky zdrojově otevřených, neboli open source, případně z programů volně ke stažení. Jedinou výjimkou je stanice s placeným operačním systémem Windows XP Professional, kterou jsem použil při testování pravidel IDS systému SNORT. Na počítači s MS Windows jsem měl nainstalován Cerberus FTP Server (Verze 3.0.8), který je pro nekomerční využití k dispozici zdarma. Kromě tohoto FTP serveru jsem na OS Windows instaloval poštovní SMTP a POP3 server s názvem hMailServer (Verze 5.3-B1617) a poštovního klienta Opera Mail (Verze 10.10) pro příjem a odesílání e-mailů. Poštovní server i klient jsou opět z nabídky volně dostupného softwaru. Na dalších stanicích jsem již využíval operační systém Linux Ubuntu. Mezi ostatní aplikace, které jsem využíval, patří IDS systém SNORT a penetrační nástroje PackETH, Nemesis a hping. Zmíněné aplikace jsou z kategorie zdrojově otevřených a instaloval jsem je právě na operační systém Linux Ubuntu.

Myslím si, že tato práce může sloužit jako úvodní materiál při zavádění IDS systému do vlastní sítě. Administrátor, který se rozhodne využít aplikace SNORT, zde nalezne popis vytváření pravidel a konkrétní ukázky. Navíc zde najde popis tří typů penetračních nástrojů, pro otestování svých pravidel, s popisem ovládání těchto aplikací. Na tuto diplomovou práci by mohli další studenti navázat tématy jako je IPS systémy, nebo IDS založené na odchylkách od protokolových norem.

7. Literatura

[1] Endorf, Carl – Schultz, Eugene – Mellander, Jim: Detekce a prevence počítačového útoku. Grada, Praha 2002. 1. vydání

[2] Lockhart, Andrew: Bezpečnost sítí na maximum. CP Books, Brno 2005. 1. vydání

[3] Petrlík, Lukáš: Jemný úvod do systému UNIX. Kopp, České Budějovice 2001. 1. vydání

[4] Northcutt, Stephen – Zeltser, Lenny – Winter, Scott – Frederick, Karen Kent – Ritchey, Ronald: Bezpečnost sítí: velká kniha, Bezpečnost počítačových sítí: kompletní průvodce návrhem, implementací a údržbou zabezpečené sítě. CP Books, Brno 2005. 1. vydání

[5] Odom, Wendell: CCENT/CCNA ICND1 Official Exam Cerfification Guide. Cisco Press, Indianapolis, USA, 2008. 2. vydání

[6] Odom, Wendell: CCENT/CCNA ICND2 Official Exam Cerfification Guide. Cisco Press, Indianapolis, USA, 2008. 2. vydání

[7] Thomas, Thomas M. – Vrátenský, David: Zabezpečení počítačových sítí bez předchozích znalostí. CP Books, Brno, 2005. 1. vydání

Elektronická literatura

[8] Bogartes, Phillipe: HPING tutoriál, verze 1.5, 2003. Dostupný z www: <http://www.radarhack.com/tutorial/hping2.pdf>

[9] Snort Users Manual, verze 2.8.4, 2009. Dostupný z www: <http://www.Snort.org/assets/140/Snort_manual_2_8_6.pdf>

[10] PackETH – paketový generátor. Dostupný z www: <http://packeth.sourceforge.net/>

[11] Nemesis – paketový generátor. Dostupný z www: <http://nemesis.sourceforge.net/>

[12] Psaní pravidel SNORTu. Dostupné z www: <http://packetstormsecurity.nl/papers/IDS/Snort_rules.htm>

[13] Čísla portů. Dostupné z www: <http://www.iana.org/assignments/port-numbers>

[14] Typy ICMP. Dostupné z www: <http://www.iana.org/assignments/icmp-parameters>

[15] Čísla portů. Dostupné z www: <http://www.iana.org/assignments/port-numbers>

[16] Konfigurace zrcadlení portů. Dostupné z www: <http://www.samuraj-cz.com/clanek/cisco-ios-22-monitoringkontrolazrcadleni-provozu-span-a-rspan/>

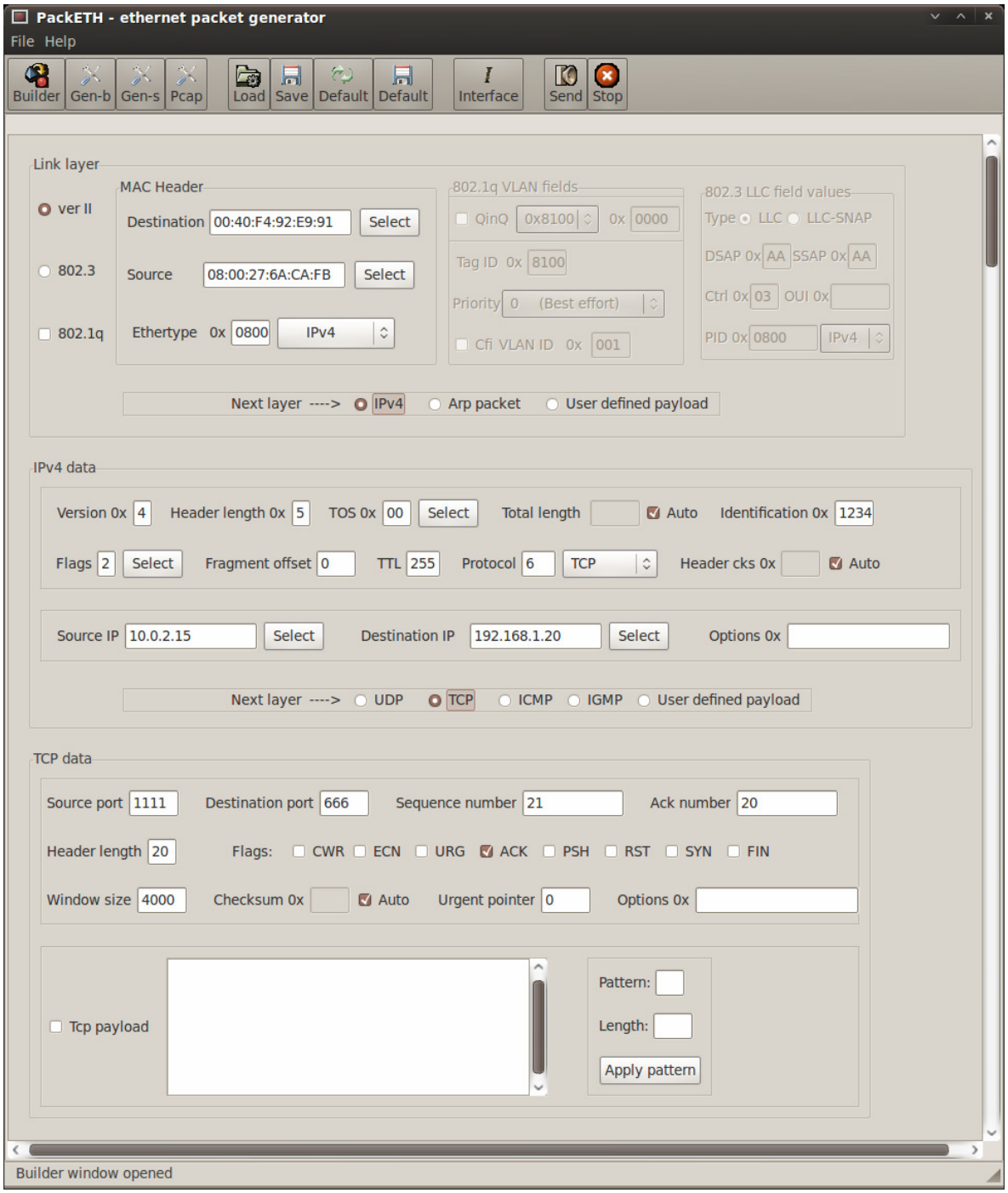
Seznam příloh:

- Příloha č.1 – PackETH generátor provozu
- Příloha č.2 – Nemesis generátor provozu
- Příloha č.3 – hping generátor provozu
- Příloha č.4 – 3 way handshake
- Příloha č.5 – Konfigurační soubory - CD

Příloha č.1 1/4

PackETH – generátor provozu

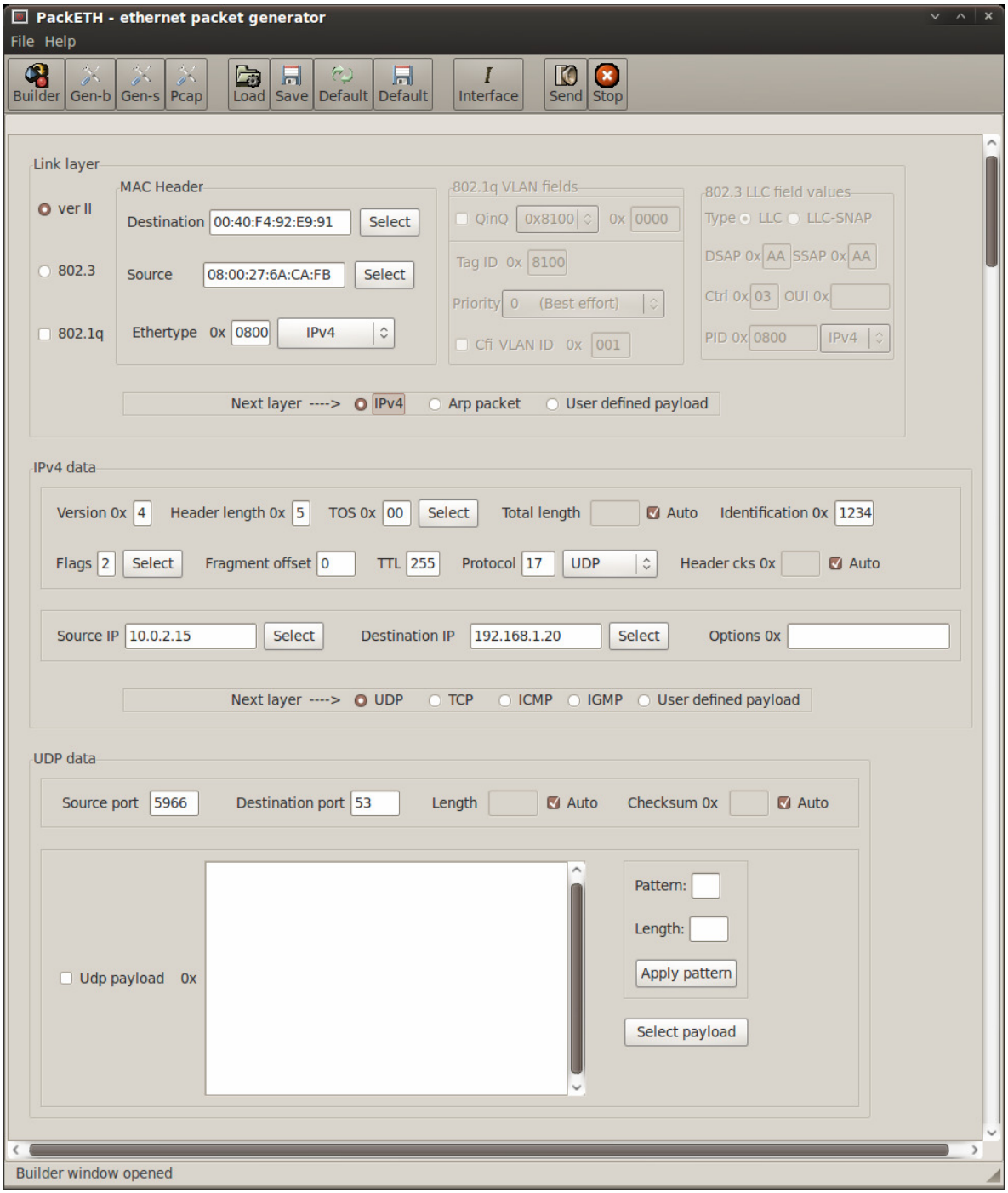
Záložka „builder“ – nastavení TCP paketu



Příloha č.1 2/4

PackETH – generátor provozu

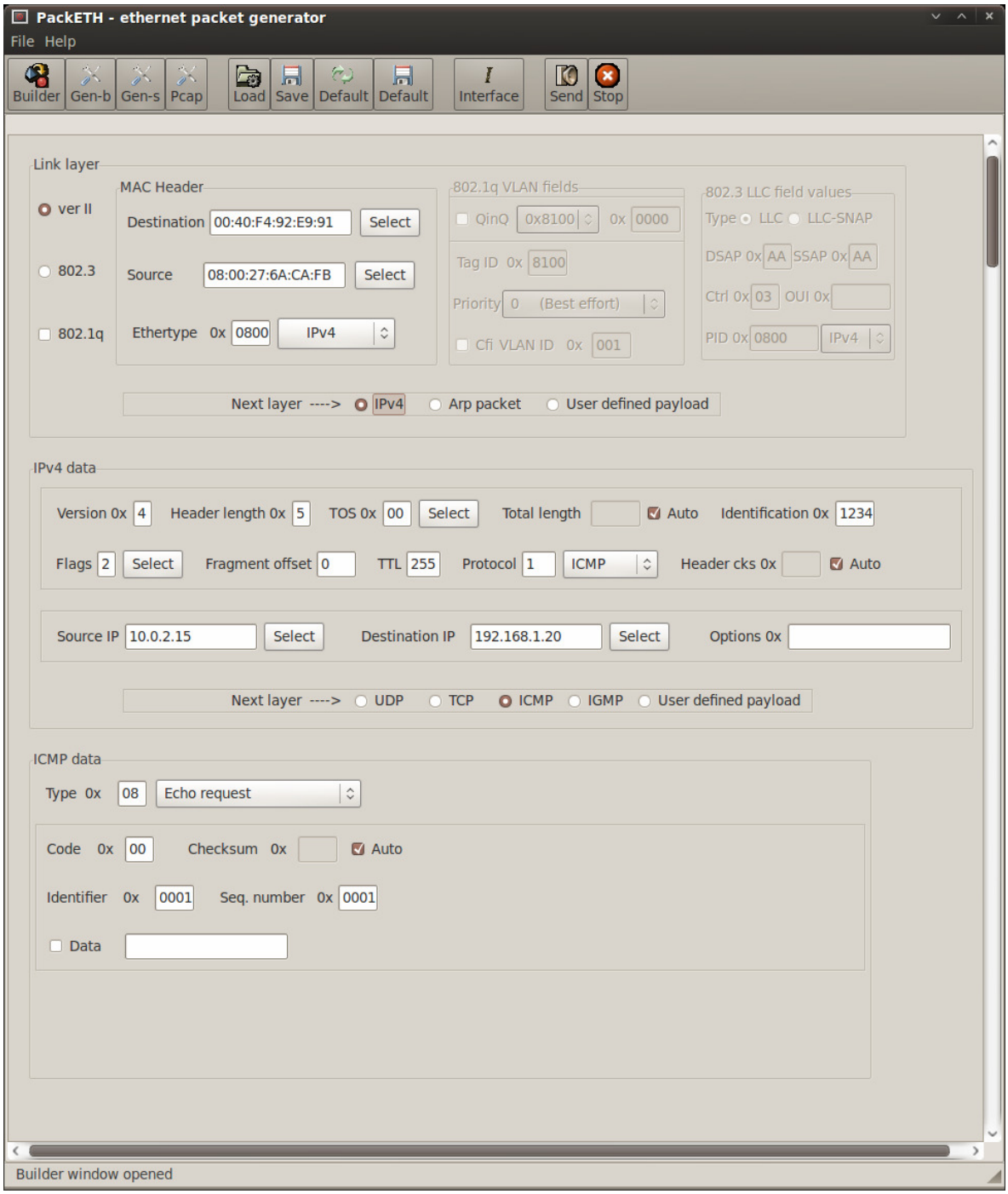
Záložka „builder“ – nastavení UDP paketu



Příloha č.1 3/4

PackETH – generátor provozu

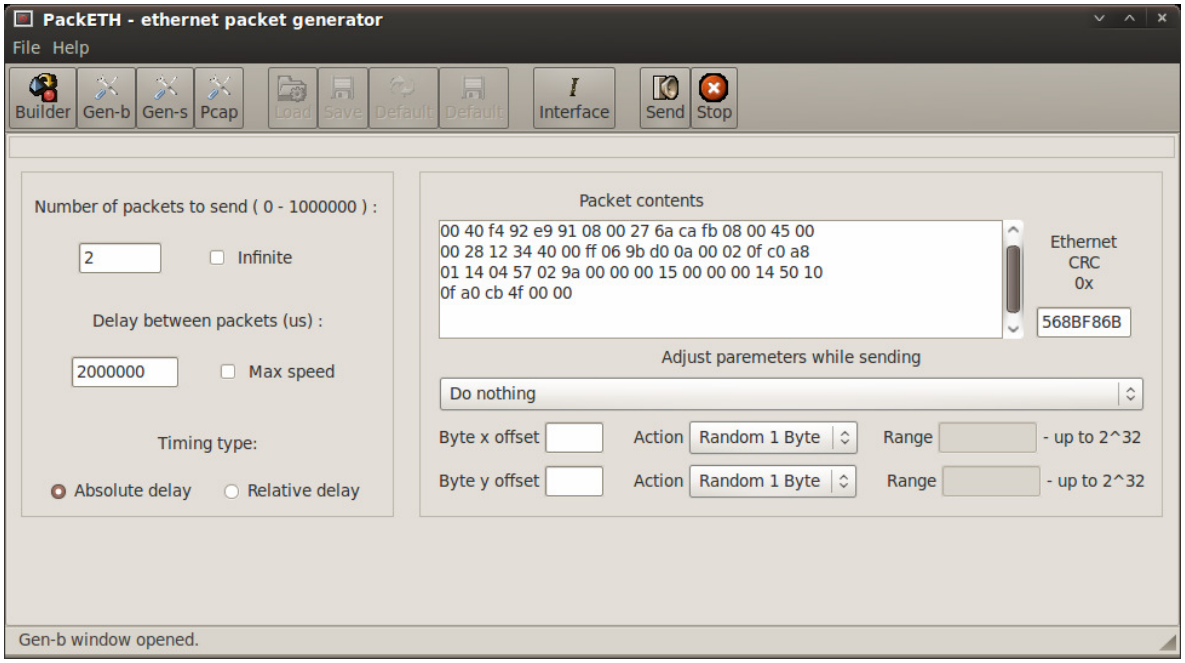
Záložka „builder“ – nastavení ICMP paketu



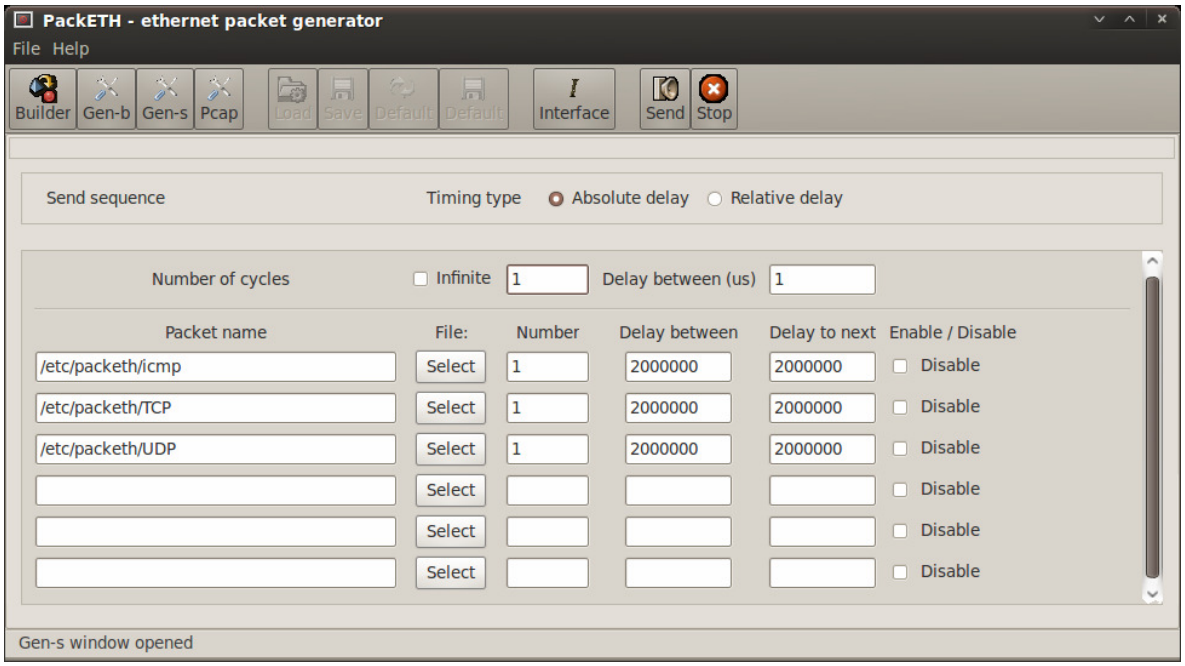
Příloha č.1 4/4

PackETH – generátor provozu

Záložka „Gen-b“



Záložka „Gen-b“



Příloha č.2 1/2

Nemesis – generátor provozu

Popis použitých parametrů pro generování ICMP paketů

```
nemesis icmp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91 -H
08:00:27:6A:CA:FB -i 8 -c 00 -qE -v
```

```
nemesis icmp [-v mód výřečnosti] [-c kód ICMP] [-d ethernetové rozhraní]
[-D cílová IP adresa] [-F režim fragmentace] [-H zdrojová MAC adresa]
[-i typ ICMP] [-m maska IP adresy] [-M cílová MAC adresa] [-P soubor užitečné
zprávy] [-q mód vyslání ICMP paketu] [-S zdrojová IP adresa]
```

-F režim fragmentace

- FD (bez fragmentace)
- FM (více fragmentů)

-q mód vyslání ICMP paketu

- qE (ICMP echo)
- qU (ICMP nedostupný)
- qX (ICMP vypršení času)
- qR (ICMP přesměrování)

Popis použitých parametrů pro generování TCP paketů

```
nemesis tcp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91
-H 08:00:27:6A:CA:FB -x 1111 -y 666 -s 21 -a 20 -fA -w 4000 -v
```

```
nemesis tcp [-v mód výřečnosti] [-a potvrzující číslo] [-d ethernetové rozhraní]
[-D cílová IP adresa] [-f TCP příznak] [-F režim fragmentace] [-H zdrojová MAC
adresa] [-I IP ID] [-M cílová MAC adresa] [-P soubor užitečné zprávy]
[-s sekvenční číslo] [-S zdrojová IP adresa] [-T IP TTL] [-w window size]
[-x zdrojový port] [-y cílový port]
```

Příloha č.2 2/2

Popis použitých parametrů pro generování UDP paketů

```
nemesis udp -d eth0 -S 0.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91
-H 08:00:27:6A:CA:FB -x 5966 -y 53 -v
```

```
Nemesis udp [-v mód výřečnosti] [-d ethernetové rozhraní] [-D cílová IP adresa]
[-F režim fragmentace] [-H zdrojová MAC adresa] [-I IP ID] [-M cílová MAC
adresa] [-P soubor užitečné zprávy] [-S zdrojová IP adresa] [-T IP TTL] [-x
zdrojový port] [-y cílový port]
```

Výpis shall souboru „mujshall.sh“

```
#!/bin/bash

p1=1

while [ $p1 -gt 0 ]
do
 {
  nemesis icmp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91
  -H  08:00:27:6A:CA:FB -i 8 -c 00 -qE -v
  nemesis tcp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91
  -H 08:00:27:6A:CA:FB -x 1111 -y 666 -s 21 -a 20 -fA -w 4000 -v
  nemesis udp -d eth0 -S 10.0.2.15 -D 192.168.1.20 -M 00:40:F4:92:E9:91
  -H 08:00:27:6A:CA:FB -x 5966 -y 53 -v

  p1=$(($p1-1))
 }
Done

echo shell odeslal pomoci nemesisu pakety a zkoncil
```

Příloha č.3 1/1

hping – generátor provozu

Popis použitých parametrů pro generování paketů

hping -h zobrazení nápovědy

-c počet odeslaných paketů

-i časový rozestup mezi rozesílanými pakety

-I rozhraní ze kterého se vytvořený paket odešle

-V mód výřečnosti

Protokol

-1 ICMP paket

-2 UDP paket

(TCP paket je v základním nastavení, není pro něj třeba zadávat speciální klíčové slovo)

ICMP

-c typ ICMP zprávy

-k kód ICMP zprávy

TCP a UDP

-s zdrojový port (náhodný)

-k neměnit zdrojový port

-p cílový port

-M sekvenční číslo

-L potvrzující číslo

-F příznak FIN

-S příznak SYN

-R příznak RST

-P příznak PUSH

-A příznak ACK

-U příznaku URG

Příloha č.4 1/1

3 way handshake